Translate For Good - Building a Human-Based Translation Service

Ekatarina Arslanbaeva, Marina Orzechowski, Natalia Harrow, Sajad Gholamzadehrizi

***Abstract*—The present paper describes the process of designing and implementing a full stack web application aiming to provide free human-based translation services to immigrants, refugees, and tourists. The application's main purpose is to connect people with volunteering translators. The process description consists of the following sections: project context, project implementation, results, evaluation, limitations, timeline, conclusions, acknowledgment, and references. The project context describes the problem, links to related work, and our solution. The project implementation describes the organization and technology used to complete the project, as well as the architectural design of the application. The results talks about the minimal viable product, use cases, and displays demo screenshots for the application. The rest of the paper describes how we evaluated the project and what limitations we had while delivering the application.**

***Index Terms*—translation services, real time chat, voice messages, image sharing.**

I. Project Context

A. Problem Description

Language barrier is a problem known to many people traveling to a country whose language they do not speak fluently. The problem might affect tourists visiting a foreign country, immigrants uprooting their families to start a new life, or refugees crossing national boundaries. Whether a person is having trouble understanding a label on a food product or struggles writing a professional email, the root of the problem is the same - linguistic barrier to communication, which even in most seemingly trivial situations can lead to frustration and the feeling of alienation.

B. Related Work

There are many tools available online providing translation services. They can be divided into two groups: non-human based and human-based apps.

Non-human based applications rely on machine translations. Some examples of such apps are Google Translate, Microsoft Translator, and iTranslate. While such applications provide quick and convenient translations, they often lack accuracy, real-human connection, and cultural knowledge.

Human-based translation services, on the other hand, while accurate and reliable are often costly. Moreover, most of such services are primarily used for translations of official documents and not in situations of a daily life. They are expensive, slow, and not available around the clock.

Interpreter.com is an example of a human-based translation services. It offers live on demand telephone interpretation services 24/7. The main downside of this service is that it is available only withing the United States. Besides that it is not free and users are charged $2.99/minute.

Tarjimly is the most successful application which provides free on demand translation services 24/7 worldwide.This mobile app allows multilingual speakers to remotely volunteer their language skills as translators. The application matches translators with people in need and connects them in a live chat, where they can send text, documents, and start a phone or video call.

While this application is very successful, it has some technical drawbacks.

- According to reviews on Google Play and Apple Store both Android and iOS versions might crash or freeze at the matching phase, verification emails are not received by new subscribers, the app is very slow.
- There is no place to look which requests are active and which are already satisfied or expired. When new translation sessions are requested matched translators receive links to a chat room via email. When the request is satisfied by a translator other translators are not notified. So they spend time preparing for a translation session (find a quiet place), click a link and find out that their services are not needed anymore.
- Users must have two accounts if they want to be a translator and a regular user at the same time. It might be annoying to switch accounts.
- The application offers translation services in many fields including legal/asylum and medical. However, translators are not tested enough to perform translation in such sensitive situations and might unintentionally harm the case.

C. Our Solution

To solve the issue of impersonal and unreliable machine-based translations and expensive and slow services hiring professional translators, we introduce a free platform connecting users needing a translation from one language to another with volunteers who speak both languages. Users can use the service during their daily activities, such as grocery shopping or visiting a hair salon, or while needing help. To best serve the users, the platform will enable them to upload different types of files needing a translation and will match them with volunteer translators in real time.

II. Project Implementation

A. Organization

The delivered full stack web application has been finalized in May 2021. In order to organize the team work throughout

the process, we utilized a number of tools to keep the team on track.

1) Agile Methodology: Our team followed the Agile practices approach while developing this application throughout the process. By being Agile, we had iterative approach in design, planning, development and testing of our application. In addition to frequent stand-ups, we kept in touch using online platforms including Trello project management, Zoom, WhatsApp, and also held regular sprint meetings every week. Our team adopted Kanban practices to be able to be more flexible with individuals schedules and be able to prioritize tasks faster.

2) Meetings and Project Management: We used Trello as the main project management tool for keeping track of the progress. Every team member was able to create and assign tasks on Trello. This process was updated on a weekly basis. Since the Academic year was held remotely, our group meetings also was held online using Zoom as the main platform.

3) Github: We used GitHub as the main platform to version control our project as a team. Here is the link to our repository for this project. TranslateForGood GitHub. This helped us avoid merging conflicts and make cooperation very smooth. We also set up a deployment pipeline on Heroku and connected the main branch for automatic builds on each update of our project.

4) Deployment on Heroku: We used Heroku as the platform to deploy our project. Here is the link to our deployed application. TranslateForGood. Deployment pipeline was set up early on in the process and helped us to continuously deliver new updates as we continuously integrated new features into the application.

5) Delivery Time: This project has been delivered in May 2021. Please see more details in the Timeline section.

B. Technology Landscape

Set of technologies that were used for this project are based on criteria such as ease of implementation and maintenance, greater access to helpful resources from the developer communities, and scalability. MERN stack gave us the mentioned advantages to build a robust and modern Full Stack application.

1) React: React is a Front End library based on JavaScript language. React is developed and maintained by Facebook, as well as a large active community of development companies and individual Software Engineers. React helped us build the Front End side of our Full Stack Web application without worrying about repeated DOM elements. It helped with efficiency and scalability of our application. Additionally, in the future if we decide to scale our application for Android or iOS devices, this will help achieve that faster.

2) Node.js: Node.js is a cross-platform JavaScript runtime environment programming language. It was initially built for Google Chrome, and later open-sourced by Google in 2008. Therefore a great community of developers contribute in this programming language to an unlimited number of open source tools. Node.js helped us build an scalable network application, and could execute our React code from the front end very efficiently. Since both Node and React utilize the same programming language, it also helped us achieve a better teamwork and consistent application.

3) Express.js: Express.js is a backend web application framework for Node.js. This package helped us simplify writing our web server code. Using this tool improved efficiency as we did not need to write repeated Node.js code for http servers.

4) MongoDB: MongoDB is a cross platform document base database. Unlike traditional databases, MongoDB allowed us to have schema-less or non-relational databases. Our stored data was maintained in flexible forms of document with a JSON based format. The structure of the stored data can change overtime, which helped us to get up to speed faster while implementing our application.

5) Socket.io: Socket.io is a JavaScript library developed to utilize open connections to facilitate real time communication in modern applications. Socket.io allowed bi-directional communication between web clients and servers. Socket.io enables bi-directional communications when a client runs Socket.io client side in the browser, and a server has also integrated the Socket.io Node.js package. This library helped simplify the real time chat of our application by establishing the real time bi-directional communication and sending the data in a simple JSON format.

6) Progressive Web Application Technologies (PWA): PWAs are websites which provide functionality that would normally require a native application. PWAs can be installed to a device's home screen without an app store. Unlike regular web apps with simple home screen links or bookmarks, PWAs can be downloaded in advance and can work offline, as well as use regular Web APIs. PWAs' main technologies are Service Workers and Web App Manifest.
A service worker is a script that browsers runs in the background separate from a web page, providing a rich offline experiences, periodic background syncs, and push notifications. The web app manifest is a JSON file that tells the browser about the PWA and how it should behave when installed on the user's desktop or mobile device - what icon and name is displayed on the home screen, what is default orientation, etc. PWA are also able to send notifications using service workers, additionally we utilized Web Push API.

C. Architectural Design

1) General User-Translator Interaction: The general flow of application include both user and translator. The user and translator interact during the translation session. Users can choose the type of translation,request translation, get matched with translators available, establish a real time chat once found a volunteer, send text, upload an image or file or send their voice message while in a translation session, and complete the translation when they are done. (figure 1).

2) Application Components: Application components include all the aspects of user-translator interaction starting from the login process up to a chat room session with a translator. The main components represented in this diagram will be shown in detail below. (figure 2).

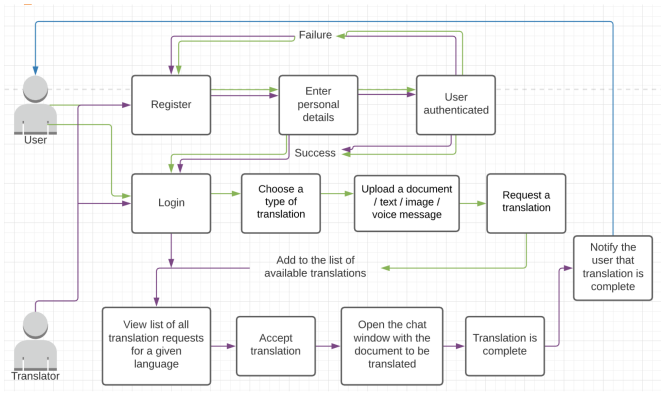

Fig. 1. User-Translator General Interaction

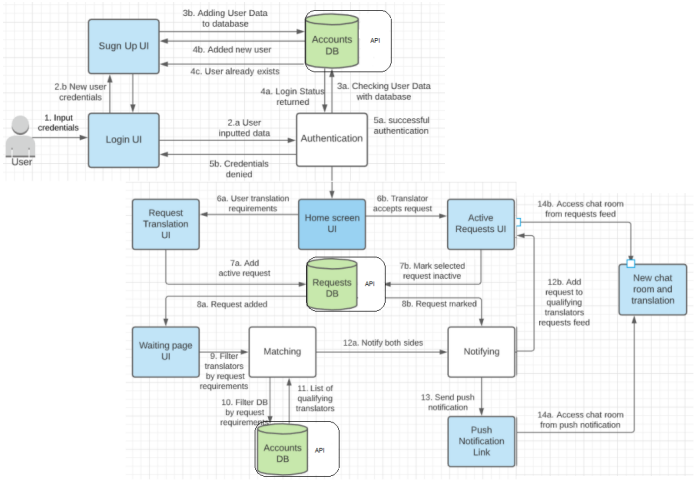

Fig. 2. Application Components

3) Sign Up Process: Person, which would like to use our application should sign up. To do that, a person needs to open a sign-up page and enter the desired email, username, password, if they would like to become a translator, they have the option to do so and choose the languages they are comfortable translation to and from, and if they are able to proofread as well, they also can identify if they are female so we can facilitate gender specified translation requests in occasions such as doctor appointment, country they live in and timezone, into the sign up form. Next, the person registered as a user and translator or regular user and is able to login (figure 3).

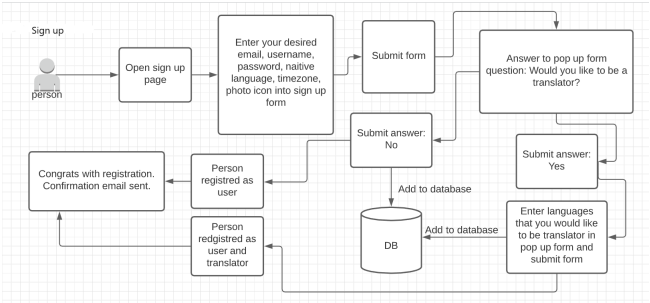

Fig. 3. Sign Up Process

4) Log In Process: Users and Translators must log in with their credentials to access the application's UI. If any user provides invalid credentials, the user will be notified and will not be permitted to access the application. (figure 4).

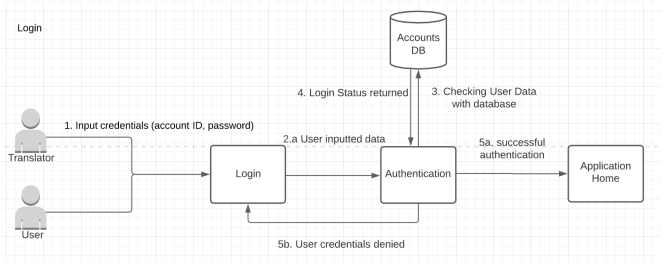

Fig. 4. Log In Process

5) Request Translation Process: User fills out the request information by specifying a pair of languages (translate from-to), and a due date of their request, and if they only want a female translator (due to cultural specificities), if they need a document proofreading (translator must be a native speaker or highly skilled in the selected language). User also must specify a due date for the translation request which cannot be earlier than an hour from the moment of submission. Then the request is added to the database as an unanswered request. (figure 5).

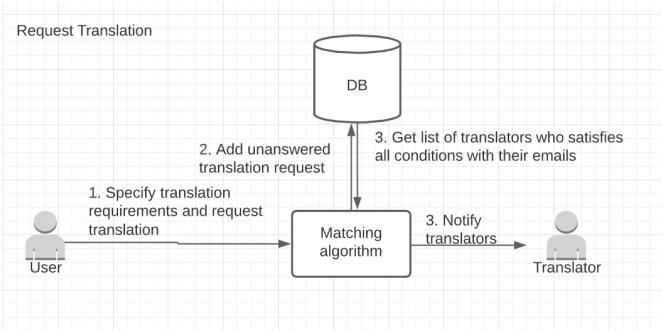

Fig. 5. Request Translation Process

6) Matching Algorithm: When user submits a request the matching algorithm starts (figure 7). First it checks if the due date for this request passed. If yes the process is terminated and user is notified about the expiration.If no, all translators are filtered by basic parameters like language, female translator and proofreading. If there are no such translators in the database, for example a very rare language was requested, the system waits until someone new registers as a translator who qualifies, or until the due date passes. All filtered translators get ranks from best match to the worst match based on scores. (figure 6).

We sort all the translators by S scores and within each S score range by utility function.

The next step is to understand if the request is urgent or not. If the due date is less than in 5 hours it is considered urgent and more translators are notified at a time to maximize the chance that someone accepts the request sooner.

Then translators are divided into batches of N people starting

S-score:

Translator's time of the day 'activity' range.

Time range	Score
Weekday: 5pm-10pm, Weekends:8am-10pm	0
Weekday: 8am-5pm	0.3
6am - 8am, 10pm-12am	0.7
12am-6am	1

Utility Function (UF):

Translator's activity score.

Depends on percent of accepted requests of translators. The app is based on volunteer work, and the translator's workload should be fair. More requests are matched to translators who didn't accept many requests yet.

activityRate = accepted/total_matched
TAS = 1 - activityRate

Time zone match score.

Depends on GMT difference between timezones of requester and potential translator.

Tz_diff = abs(GMToffsetTranslator - GMToffsetRequester);
TZS = (26 - diff)/26;

Fig. 6. Scores System for Ranking Translators.

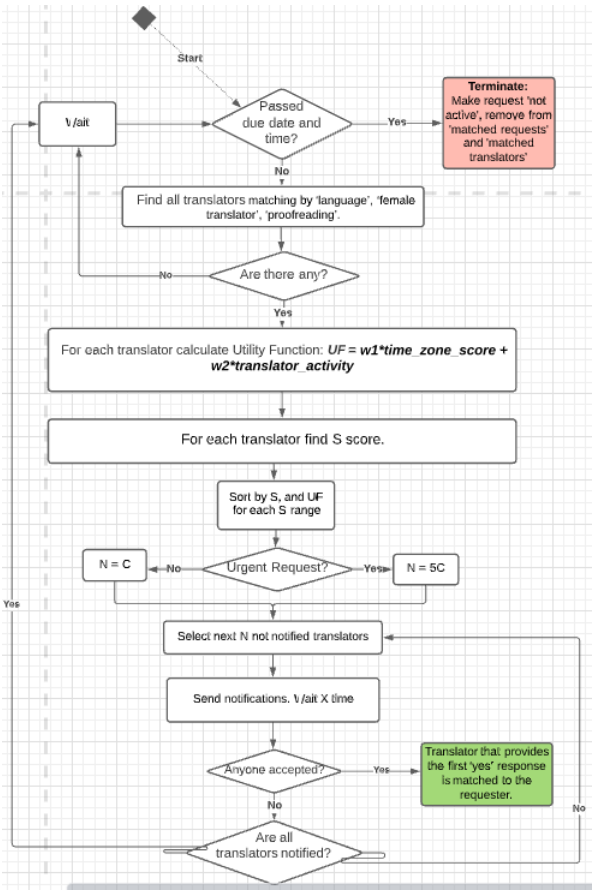

Fig. 7. Matching Algorithm Diagram.

with best match. The first batch gets notified and system waits and checks if anyone accepts the request. If yes, the process stops. If no, either the next batch gets notified, or, if everyone is already notified, system waits for someone new to register or the due date to pass. Matched translators are notified via push notifications as well as users whose requests expired or were accepted. The user waits for the translator to accept the request and once accepted, they can find the corresponding chatroom and enters the chat window, start the conversation or uploads necessary files/sends messages.

7) Translation Process - Chat Component: Once a translator accepts a translation request, a new chatroom is created. The chatroom name consists of translator's name, user's name and the languages "to and from" of the translation to make it easy for the users to navigate to an appropriate chatroom. Once the chatroom is created, both the user and the translator can initiate the conversation. Users and translators have a variety of message formats available in the chatroom to choose the most convenient one for the given translation requests. They can choose to send a text message, image, voice message and a PDF document. Once the user provides the content they need translated in the form of their choice, the translator can perform translation. When the translation is complete, the translator sends the completed translation, again choosing the most convenient form for the given request. Once the user reviews the translation, they communicate feedback to the translator. If the user needs more help with the given requests or has some clarifying questions, they can continue the conversation with the translator. If they are satisfied with the translation, they can click "Finish Translation" button displayed in the chatroom (Fig. 8). Once that happens, the

Fig. 8. Finish Translation Button

translation is marked completed and the chatroom is being deleted from user and translator's chatroom list.

Translators' chatroom list is divided into two sections - "Get Translation" and "Translate" (Fig. 9). The first one

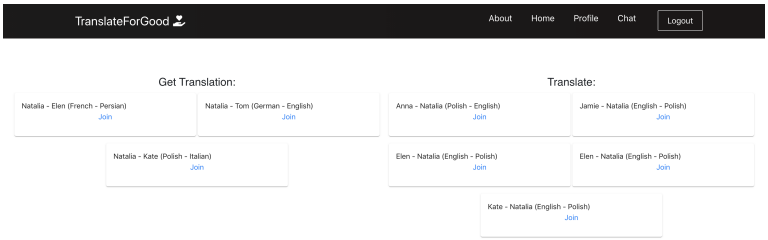

Fig. 9. Chatroom List

includes all the open chatrooms for the requests the translator submitted, and the latter contains all the chatrooms created for the requests the translator accepted and is willing to translate. (figure 10).

III. Results (MVP)

We were able to successfully implement the complete MVP as we planned in the design phase. All the fundamental features were implemented according to our timeline and we were able to develop and deliver the exact functionality we were hoping our application would provide. The complete and fully working application has been deployed and can be found at this link.TranslateForGood.

This section provides the details into our MVP, sample use cases and the final product.

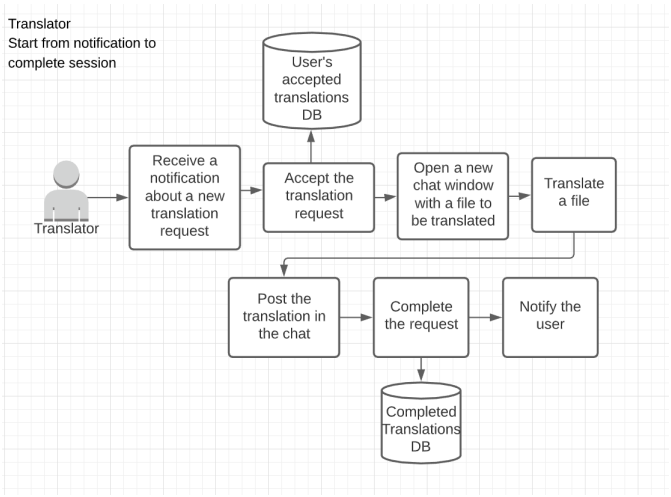

Fig. 10. Translation Process

A. Minimum Viable Product

1) **Sign Up**
2) **Log In**
3) **Profile Page**
   a) View related user/translator information
4) **Main/Home Page**
   a) User Side
      i) Request a translation session
      ii) Matching with translators of the desired language
      iii) Notify the set of translators with given criteria via push notifications
      iv) The chat room for user and translator is created after the first translator accept translation request
   b) Translator Side
      i) Notified by web push notifications
         A) Click on notification to be taken to application page
      ii) If already logged in, skip, else take them to the login page
      iii) Enter the session (chat window)
      iv) Start a session that is a one-to-one connection for users to chat
         A) Receive document, text message, voice message or image from user
5) **User-Translator Chat Window**
   a) Features to send files (document, photo, image), text messages and voice messages.

B. Use Cases

1) **Grocery store:** User is at a grocery store, products in a second language, and looks for an available translator on the app.
   a) IMAGE
      i) The user sends an image of the product to the translator in the chat window by using an image or photo features.
      ii) The translator replies with translation in the chat window by text message.
2) **Cover letter Proofreading:** User wants to send a recruiter their cover letter notifying them they're interested in a job and needs help writing a polite and professional email.
   a) TEXT
      i) The user sends the pdf file of the cover letter he/she wrote by attaching it in the chat window.
      ii) The translator proofread the file to make sure the cover letter sounds polite and professional and send it back to the user by attaching the updated file.
3) **Hairdressing Salon:** User wants to ask the hairdresser to cut their hair in their desired style. The hairdresser might have additional suggestions and ideas.
   a) VOICE
      i) The user sends a text or voice message with a description of the desired haircut and other preferences in hairstyle for his/her hair salon appointment.
      ii) The translator replies with translation in the chat window by text message or voice message.
      iii) The user shows the text with translation to the hairdresser or plays the voice message with translation to the hairdresser.

C. Demo Screenshots

Application main page will have a translation session request. The user can select languages from the drop-down lists: From and To. Additional choices provided for adue date of their request, the only female translator, and document proofreading. On top of the menu page, the user can click on About to go to About page to read information about the application and click on Profile in case they want to update their information or become a translator/regular user at any time they prefer. (figure 11).

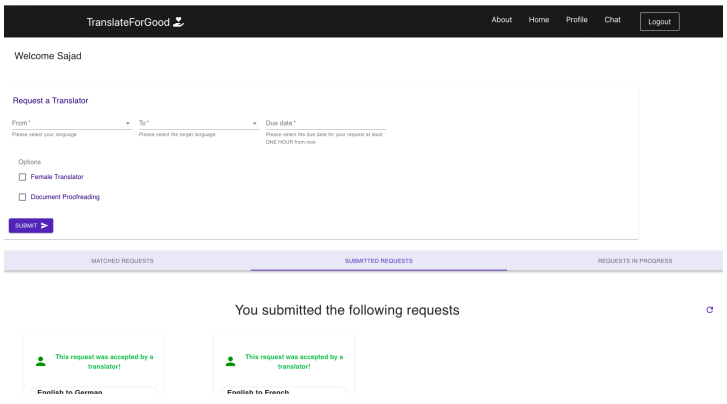

Fig. 11. Home Page

In case user intends to update their profile page they are able to click on Profile and update their information on profile page. (figure 12).

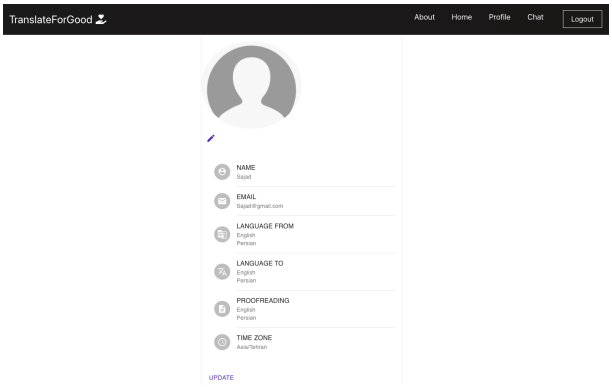

Fig. 12. Profile Page

After the user hits the Submit button on the home page, the application starts its matching algorithm and notifies the set of translators. This process will take some time. A popup appear to let the user know that the request is successfully submitted and user is notified that they can safely wait for a matched. After which, they request appears under the submitted requests tab. (Figure 13).

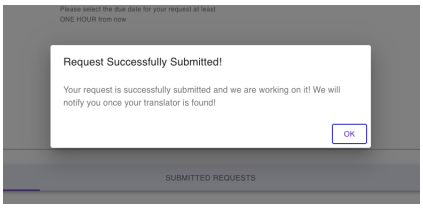

Fig. 13. Request successfully created

When the translator accepts a translation request, the author gets notified(figure 14), and a chat of translation session will be opened. The chat has the names of the translator and user in the session and languages of the translation session. In a bottom chat bar, the user can send the document, photo from camera, image from device, voice message or type text in the chat window. (figure 15).

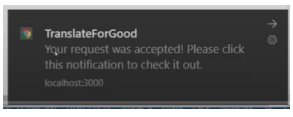

Fig. 14. Notification send to a user.

When translation is complete user can close the chatroom by clicking on ”Finish Translation” button. Translator can see the updates in their chat in real-time. They are not able to send any more messages and can only leave the chatroom (figure 16).

Additionally, we have prepared a recording of complete demo of the three use case stories of our application along with brief description of features. The link to the video is found here. Demo Video.

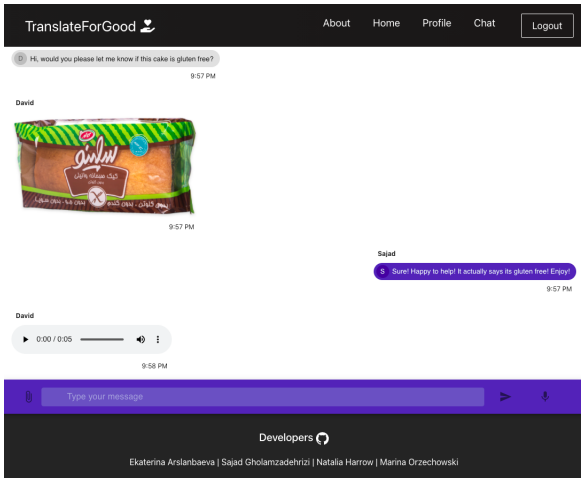

Fig. 15. Chat Page

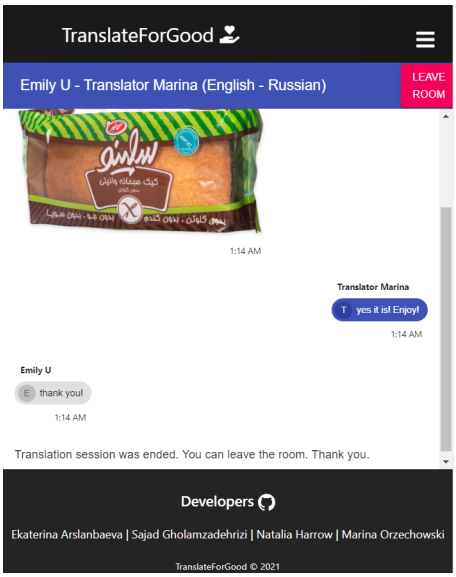

Fig. 16. Finished Translation Session Chatroom

IV. Evaluation

A. Evaluation

Our progress and success was measured by the number of the planned features that are implemented and use cases those features will cover. To evaluate our progress we used an online tool, Trello, to record, assign, estimate, categorize and analyze weekly tasks. By the end of the semester we were able to successfully complete and test all the created cards.

In the previous sections we demonstrated correct work of the chat component and the notification system.

The next component that we tested was the Matching Algorithm. We registered 30 translators speaking the same languages (English and Persian) and living in different timezones. Some of them accepted more requests than the others. Finally, we created a translation request and confirmed that translators were ranked and notified according to the matching algorithm scoring system (figure 17).

Next, we wanted to test if our application satisfies Progressive

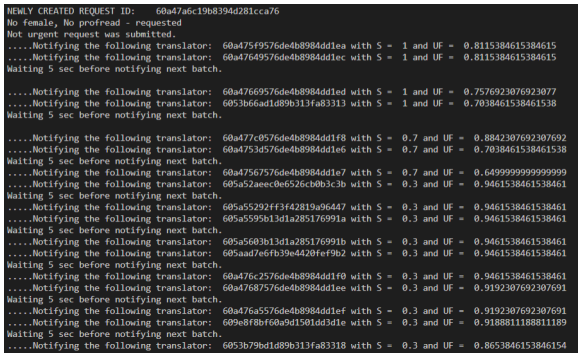

Fig. 17. Evaluation of the Matching Algorithm

Web Application requirements. To test it we run the Google Chrome Lighthouse Audit and confirmed that our application is indeed progressive (figure 18).

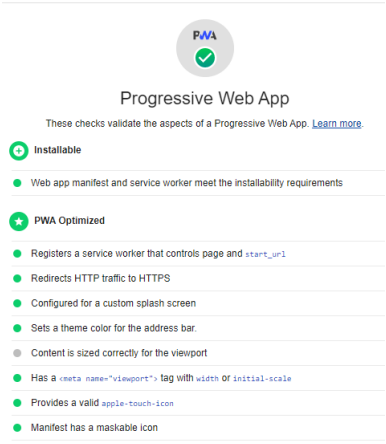

Fig. 18. Progressive Web Application Evaluation

One of the features of PWA is an ability to be installed on a mobile devices homescreens (figure 19).

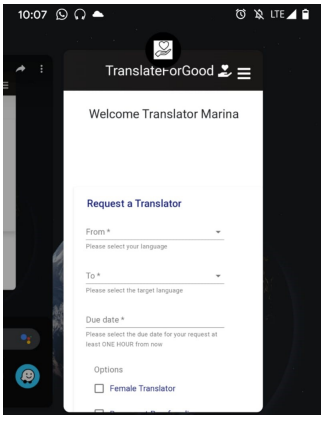

Fig. 19. PWA Feature: Installation to the Mobile Device Homescreen.

We also tested and confirmed that elements of the app shell (static elements) work offline providing a better user experience.

We were able to create a web application with great overall performance, accessibility, it is optimized for search engines, and uses best practices(figure 20).

By the end of our timeline, we were able to successfully

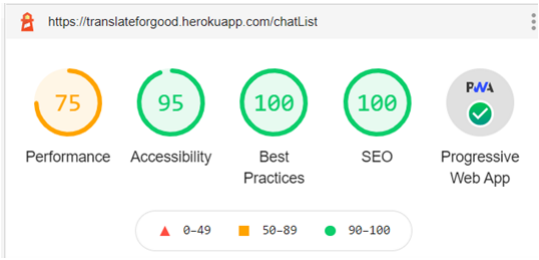

Fig. 20. Google Chrome Lighthouse Audit results

evaluate our progress by having tested all use cases initially planned for our application. The result project meets our expectations since we managed to complete all the features and use cases.

V. Limitations

A. Limited PWAs features

Our application satisfies some PWAs features such as sending push notifications with service workers. However, some features are to be implemented in future (offline support, proper installation of application on a home screen of mobile devices).

Some browsers currently don't support some features of PWAs. For example web app manifest which provide ability to install an app to a device's home screen is not supported in Internet Explorer, Firefox, Opera, it is still tested in Safari, and only has partial support by iOS Safari (figure 21).

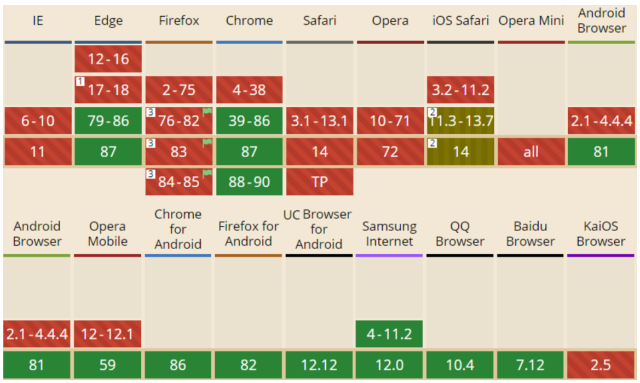

Fig. 21. Web Manifest Browser Support

Service workers which are necessary to send push web notifications are not supported in Internet Explorer and Opera Mini. It is still tested in Android Browser and Opera Mobile. (figure 22). The best browser to use for PWAs is Chrome. It supports all features and guarantees a rich user experience.

B. Notification System limitations

In order to be notified about the requests' status and new matching requests every user must agree to receive push-notifications. In case when the user does not agree, they still

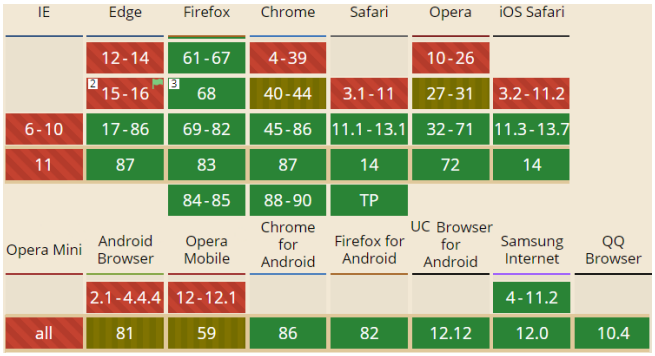

Fig. 22. Service Workers Browser Support

are able to use the application but have to manually check requests tabs for the updates.

If multiple users attempt to use the application from the same device, only one of them will be able to receive notifications. Push tokens (notification subscriptions) are bound to a device and are the same for each user. We didn't want a logged in user to receive notifications meant for someone else, therefore we limited notifications to only one user per device.

Currently each user can receive notifications on a single device. In future we will restructure notification subscription schema in a way which will allow users to receive notifications on multiple devices.

C. Attaching files to chat

The application supports attaching images, voice recordings, and PDF files. Currently other types of files are not supported. In case when the user needs to send an unsupported file, a possible solution is to use a third-party services, for example upload the file to Google Drive, create a shareable link, and send the link via TranslateForGood's chat.

D. Scheduling of sessions

Currently Translate For Good does not support scheduling of translation sessions. This is something we will work on in future.

E. Profile Pictures

Currently, users are able to only upload the profile pictures up to 32 KB, which is determined by the current format of the images stored in the database.

F. Legal aspect

There might be situations when someone needs help with their legal/asylum case or medical consultation. Our app does not test translators for their knowledge of legal/medical terms, so it is better to ask user to find certified translator.

VI. Timeline

We began working on the project on September 2020. The first stage included the design phase that we finished by the end of December. The planning and design phase included recognizing the problem we wanted to solve with our platform, defining the product users, selecting the Minimum Viable Product, creating the application architecture and flow diagrams, recognizing limitations and proposing the evaluation and mitigation plans.

We continued the implementation phase the next 15 weeks working on the project, and were able to stay on our initial timeline to finalize the project by the deadline in May 2021. Below, is a summary of the timeline for developing the application that we planned during the design phase, and successfully followed throughout the implementation phase.

1) **Week 1 and 2 (February 1 - February 15)**
   As the first step in the implementation phase, we implemented the authentication component to allow platform users to register and log in.
   a) **Authentication Component**
      i) **Front End**
         - **Registration Page**
         - **Login Page**
      ii) **Backend**
         - **User Schema**
         - **Authentication**
   b) **Tests**
2) **Week 3, 4, 5 and 6 (February 15 - March 15)]**
   After implementing the authentication process, we added basic page components like Home Page, Profile Page and About Page. We implemented the features allowing users to create requests - Request Page consisting of a form that users fill out to submit a request, and the component informing users about a successfully created request. We also implemented the Matched Requests Tab in which translators can see all the requests they have been matched with, the Submitted Requests Tab where all users can monitor the status of their requests, and the Requests in Progress Tab allowing users to see all their requests that have been accepted by a translator and all the requests they have accepted if they are a translator. We also completed the work on the matching algorithm - the integral part of our application responsible for matching users with translators.
   a) **Basic Page Components**
      i) **Home Page**
      ii) **Profile Page**
      iii) **About Page**
   b) **Requests' Components**
      i) **Front End**
         - **Request Form**
         - **Request Submitted Component**
      ii) **Back End**
         - **Request Schema**
   c) **Requests' Tracking Tabs**
      i) **Front End**
         - **Matched Requests**
         - **Submitted Requests**
         - **Requests in Progress**

ii) **Back End**
- **Retrieving requests' details**

d) **Matching**

i) **Back End**
- **Filtering translators based on basic request attributes**
- **Scoring the translators**
- **Sorting the translators by the score**

3) **Week 7 and 8 (March 15 - March 29)**
Once we implemented a system that matches users with translators, we worked on developing a notification system so that matched translators could be notified about the translation request waiting for them and users could get notified when someone accepted their request.

a) **Notification Component**

i) **Backend**
- **Service Workers**
- **Web-Push API**
- **Subscription Schema**

b) **Tests**

4) **Week 9, 10, 11 and 12 (March 29 - April 26)**
After completing the notification system, we worked on developing a chat component allowing users to communicate and exchange messages in a variety of forms - text, audio and image messages as well as PDF documents.

a) **Chat Component**

i) **Front End**
- **List of active chatrooms**
- **Chat window with file attachment and voice message recording**

i) **Backend**
- **Socket.io enabling live chat**
- **Chatroom Schema**
- **Message Scheema**

b) **PWA features**
- **Static and dynamic caching**
- **Automatic redirection from http to https**
- **Service Workers adjusting**

c) **Tests**

5) **Week 13 (April 26 - May 3)**
After implementing all features thoroughly tested the application, improved the interface and completed all non-urgent tasks listed in our Backlog.

a) **Final Test and Refinements**

6) **Week 14 and 15 (May 3 - May 17)**
Finally, in the last two weeks we ensured that all the features of our application work on the deployed version. We also prepared the final paper and demo presentation thoroughly describing the implementation phase and the final features.

a) **Finalize and Delivery of Product**

b) **Presentation and Paper**

CONCLUSION

TranslateForGood is a human-based translation application, which uses non-profit volunteers to help people translate from languages they do not know. During the second semester of our senior design class, we completed all the features we are planned to complete according to our MVP. First of all, a user can register in the app as a user or as a translator by providing languages he/she can speak and proofread. On the main page of the application, the user can request a translator. App has a matching algorithm to find good match translators for every request and notify them. The matching algorithm is one of the most important app features. It notifies translators in the best matching score order. If someone accepts a request or user, who submitted the request, cancels it or request due day and time passed, the matching algorithm for particular requests stops and notify the user about request status. The algorithm looks for translators which satisfy the request parameters, then it identifies what time of the day in the translator timezone and gives a score to the translator based on it. Also, it's looking for a translator activity score. The sum of these two scores and the deadline of request will identify the scope of translators to notify. This algorithm makes the system more fair and comfortable to use for our volunteer translators. Translators can accept/decline requests. The home page includes cards which display matched request and request in progress for translators and submitted request and request in progress to the user. The application also has the option to update personal information or become a translator for a regular user. And last but not least, the app has a chat component to connect users and translators. Chat has features to send messages, send files, send photos, and voice messages. The complete application deployed to the Heroku service. Therefore, an application is an amazing source to get translation help and for people who need a good opportunity for volunteer work.

In order to improve the application work and make it more comfortable for users and translators, the scheduling translation session feature would be good to implement. This might increase the number of application users and volunteers. To make the matching algorithm even more precise we are planning to implement a rating and reviewing system. These are just a few of many features which can be done in the future to improve the quality of our application. One of the feature's limitations is supporting only the pdf file format in chat. We are looking forward to improving that and support more file formats.

The Translate For Good application is one of our major projects which we are planning to add to our resumes as examples of excellent application design, development, teamwork, and time management skills.

ACKNOWLEDGMENT

The authors are grateful to Professor Huy Vo for the fruitful discussion and the guides over the semesters.

REFERENCES

[1] Wikibhups. "MERN (Solution Stack)." Wikitia, Wikitia, 12 Oct. 2020, wikitia.com/index.php?title=MERN%28solutionstack%29.

[2] LePage, Pete, and François Beaufort. “Add a Web App Manifest.” Web.dev, 5 Nov. 2008, web.dev/add-manifest/
[3] LePage, Pete. “How to Provide Your Own in-App Install Experience.” Web.dev, 14 Feb. 2020, developers.google.com/web/fundamentals/engage-and-retain/app-install-banners/