FLoS: Fix Localization System

**(Supporting the Human Values of Universalism
Via the Localisation of Android Apps)**

ICSE SCORE 2023

Summary Report

Abhijit Paul

Mohsin Ibna Amin

Shartaj Sajid Nahid

Shafiq-us Saleheen

Rupali Tasnim Samad

Hasnain Iqbal Shirsho

January 13, 2023

List of Contents

List of figures

1 Introduction

The seventh edition of the Student Contest on Software Engineering (SCORE) is part of the 45th International Conference on Software Engineering (ICSE 2023) which aims to promote and encourage software engineering in universities worldwide.

Our team consists of six undergraduate students from Institute of Information Technology, University of Dhaka. We have selected “Supporting the Human Value of Universalism via Localization of Android Apps” which is a project for detecting localization issues in the android applications and providing a fix. We chose this project because it will allow apps to be adapted to different languages and make it accessible. When an app is accessible, it reaches the widest possible audience and can be used by people with diverse abilities such as visual or auditory impairments, as well as those who may have difficulty with language or cultural barriers.

This report contains requirement analysis, architectural design, trade offs and choices, management plan, source code management, time management, implementation and platform choices, verification and validation activities, challenges faced and lessons learned.

2 Requirements Analysis

2.1 Overview of FLoS

The “Fix Localisation System” (FLoS) is a web application that automates the testing process for detecting localization in a mobile application and provides fixes for the issues. It handles localization for both text and media content (audio, video & image). It has three modules which are registration & login for creating an account, project handling module for creating projects & controlling access to it and finally test & fix localization module. The modules are described below:

2.1.1 Registration & Login

A user needs to sign up for creating an account and to do that, he needs to enter his username(unique), email, password and additional information (e.g company name). To verify the email address, an OTP is generated and sent to the mail. The user needs to enter that OTP to confirm his email address.

If the user forgets his password and wants to recover his account, an OTP is generated and sent to the email address for that account. The user needs to enter that OTP to authenticate his identity and now, he can create a new password for his account. Note that, the users can change their user information anytime after logging in.

2.1.2 Project Handling

After logging in, the user can create a new project and become the project owner. He may search and add other users in the project and the added users are called testers. The project owner can edit project information and view results of all localization tests run. He can set the project to be private or community. For community projects, the tester can see the test results of other testers associated with the project. For private projects, they can only view their own results.

A user can also view the list of existing projects he has access to. After selecting a project, the user can view the list of localization tests and their results for that project. The list includes the following information - the apk file that was submitted for testing, time of when the test was conducted, number of localization issues detected and fixed, fixed apk file and any additional comments.

2.1.3 Test & Fix Localization

The user can choose to run a new test for a project and for that, he uploads the apk file. Then the user needs to select languages(e.g. French, English etc) for which he wishes to run the localization tests.

After selecting languages, the user confirms his submission and then, the system initiates testing of localization that respects the value of universalism.
The system checks for localization in text and media content. It detects the language for text-content and subtitles, voice speech & captions for media-content.

After that, the user can view the Localization Test Report. The localization test report includes - list of localization issues & their content type (text or media content), status (whether issues are found or not), line of code where the localization issue has occurred, link to the content (image, audio, video) with localization issues and tester who has run the test.
After viewing the report, the user is asked to select the localization issues he wants to resolve from the localization issues' list. The system then tries to generate fixes for those localization issues in 3 steps:

1. **Translation:** System translates & saves the texts, subtitles and captions in a *translation file*. The user may check the translation file and see whether the translation is correct or not. If he does not like any translation, he may download, edit and upload the translation file.

2. **Fixing Graphical User Interface(GUI):** If translated text requires more space than the size of the GUI element the text is in, then the GUI element is resized by the system in a manner that the fixed GUI remains consistent with the original GUI.

3. **Media Content Fix:** System adds translated subtitles to video and audio in the selected language. It also translates extracted texts from images as captions to fix that issue.

After patching the fixes in the source code, the system presents the results as a Localisation Fix Report. The report includes - report version and type, issue status (total number of fixed localization issues &

unfixed localization issues), list of localization issues, their content type, line of code & whether they were fixed or not, link to the original & fixed content so that the user may view them. Tester who has run the test can download the fixed apk file or add comments once the fix generation is completed.
A notification is sent to the project owner after the localization fix report is generated. Note that the report & fixed apk file are stored in the system as a log. After a year, the project owner will be asked whether he wants to archive the log data or not.

2.2 Assumption

1. An APK file for the mobile application will be available.
2. System will generate the translation but ultimately, uploading the correct translation is the duty of the tester.
3. API for all phases of coding will be available and the API query limit will not be exceeded.

2.3 Scope

1. We only cover text & media content based localization that respects the value of universalism.
2. When generating a fix, we only consider height, width & spacing.

2.4 Requirement Specification

2.4.1 Use Case Diagram

The following use cases show the list of events that take place between the users and the system to accomplish the goal.

Level 0: FLoS System: Fix Localization System
Primary Actors: Tester, Project Owner, User
Secondary Actor: Email Service

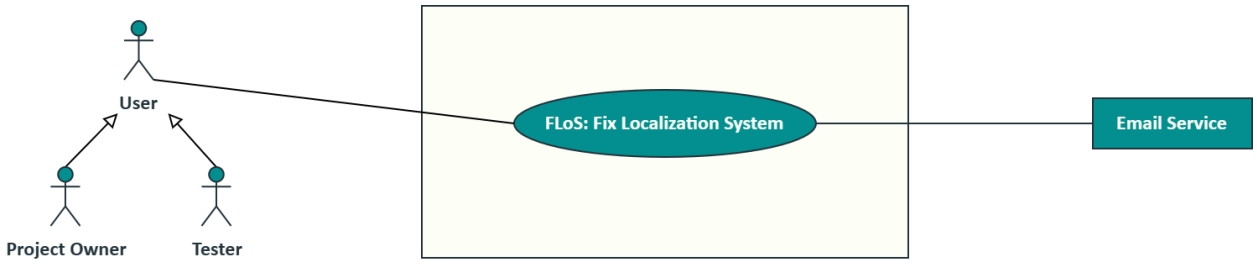

Figure 1: Use case diagram of Level 0

Level 1: FLoS Modules
Primary Actors: User, Tester, Project Owner
Second Actors: Email Service

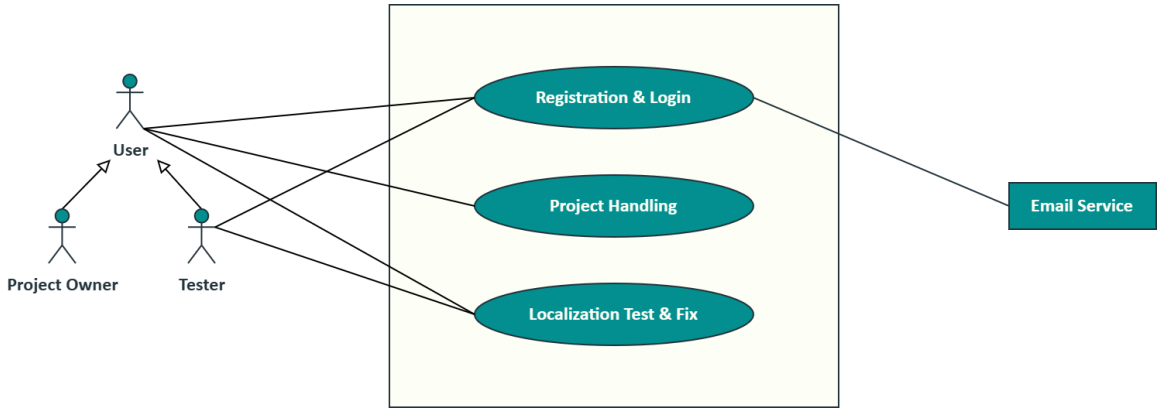

Figure 2: Use case diagram of Level 1

There are three modules in the application.

Level 1.1: Registration & Login
Primary Actors: User, Tester, Project Owner
Secondary Actors: Email Service

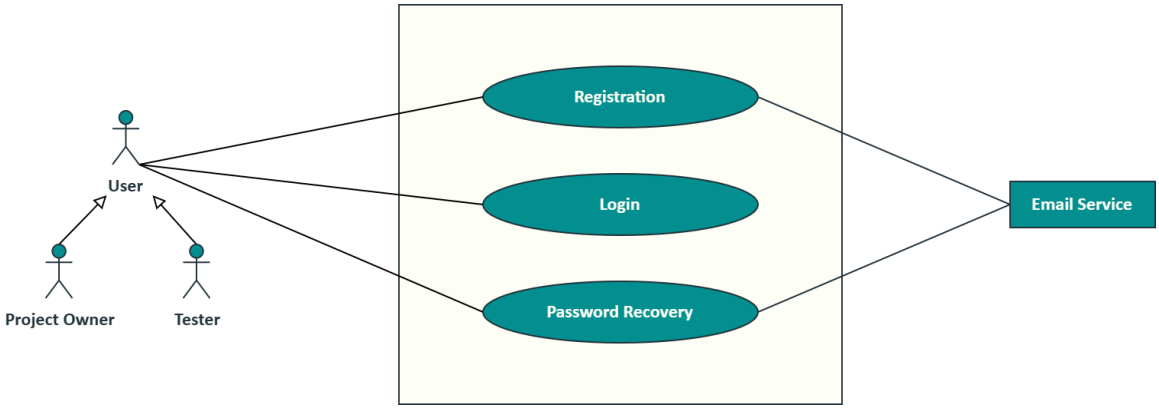

Figure 3: Use case diagram of Level 1.1

Action: New Users enter username, email id and password to create an account.
Reply: If the request is valid, the applicant will receive a confirmation email.
Action: Authenticated users enter email address and password to login.
Reply: He is allowed to enter into the system upon providing correct credentials.
Action: Authenticated users want to reset passwords.
Reply: System allows to reset the password through email.

Level 1.2: Project Handling
Primary Actors: Project Owner
Secondary Actors: Tester

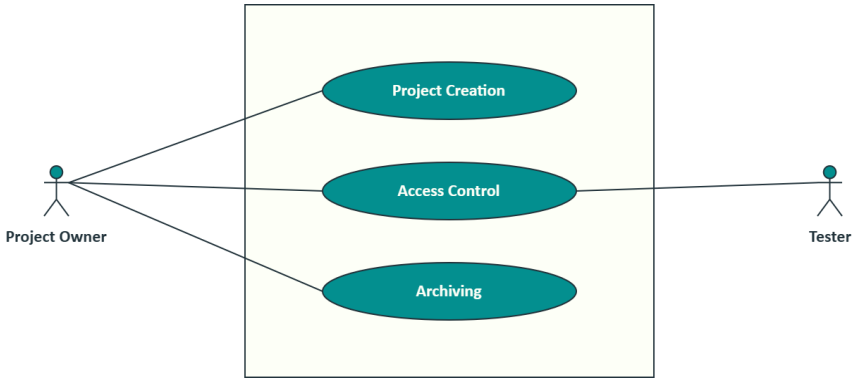

Figure 4: Use case diagram of Level 1.2

Action: The user creates a project, adds other users to the project and can edit project information.
Reply: The newly added users get notification and become a tester.
Action: The user views the projects he is part of.
Reply: User sees project's localization testing results he has access to.
Action: The project owner sets the archive time period.
Reply: Project owner is notified to archive the content when the time period is over.

Level 1.3: Localization Check & Fix
Primary Actors: User, Tester, Project Owner

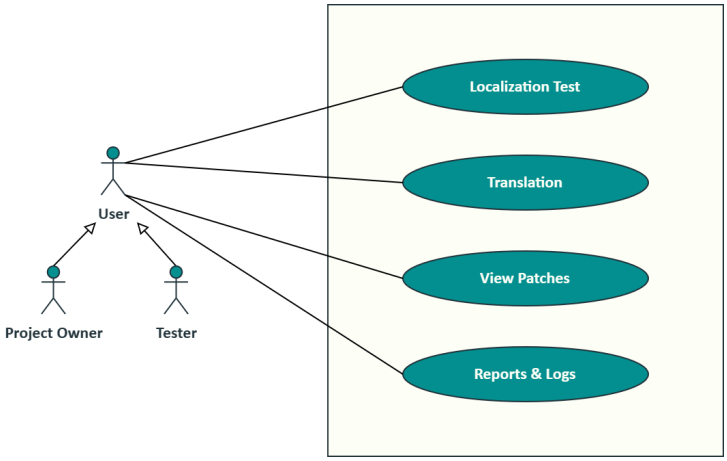

Figure 5: Use case diagram of Level 1.3

Action: User starts localization test after uploading the apk file.
Reply: Localization test completes & the user selects the issues he wishes to resolve.
Action: User downloads the translation file for the selected localization issues.
Reply: User uploads the translation file after he edits the translations.
Action: User views the localization fix report after the localization issues are resolved.
Reply: User downloads the fixed apk file.

2.4.2 Activity Diagrams

Activity diagram represents the complete flow of a particular use case. Figure 6 represents the activity diagram of level 1.1 Registration & Login Module.

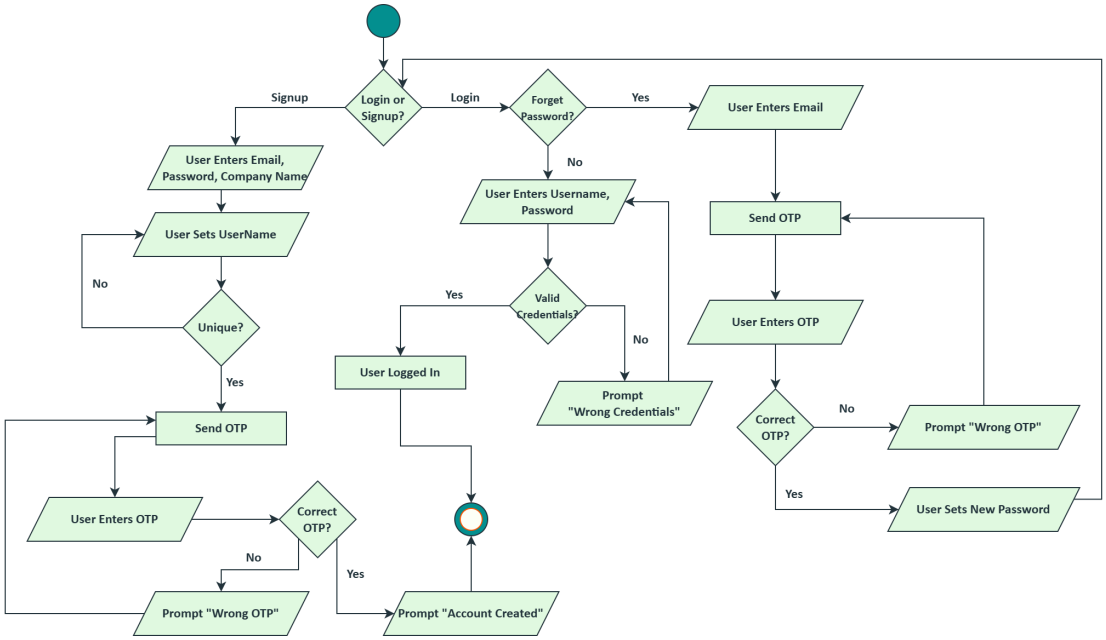

Figure 6: Activity Diagram of Registration & Login Module

Figure 7 represents the activity diagram of level 1.2 Project Management.

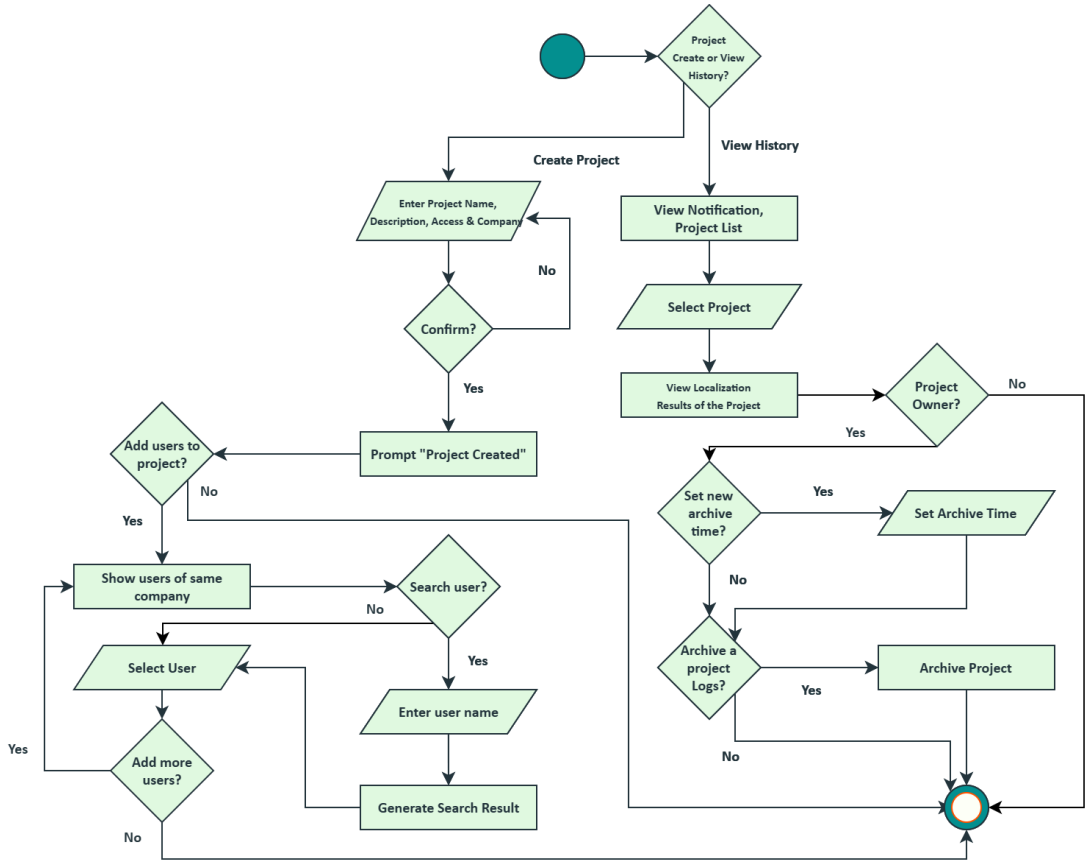

Figure 7: Activity Diagram of Project Handling

Figure 8 represents the activity diagram of level 1.3 Localisation Check & Fix.

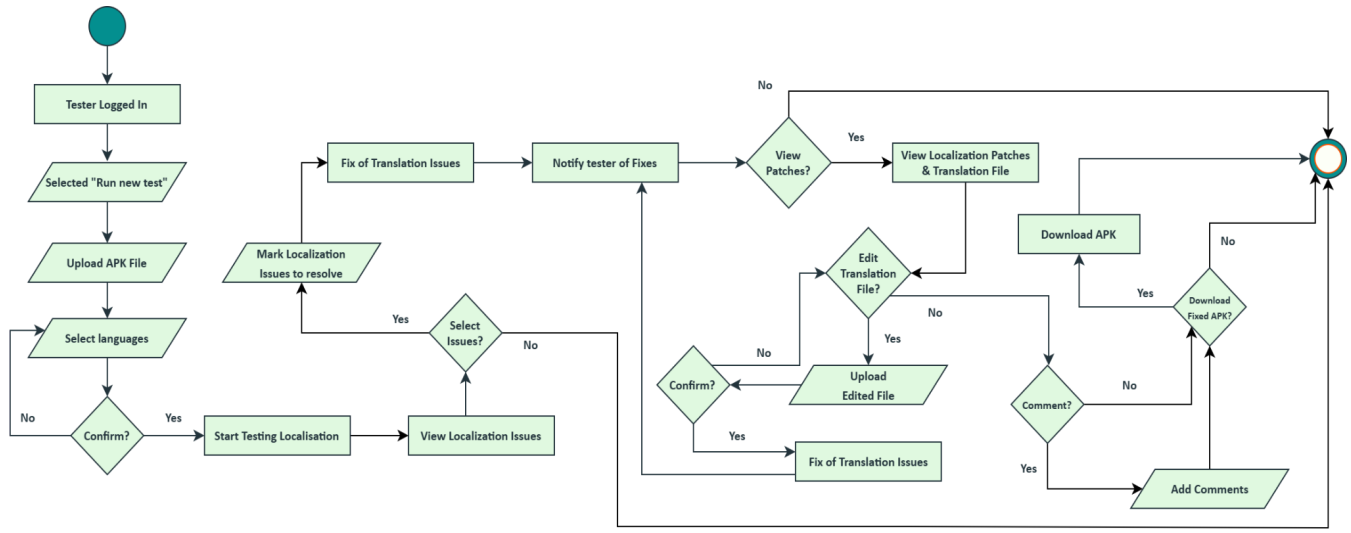

Figure 8: Activity diagram of Localisation Check & Fix

2.4.3 Entity Relationship Diagram

Figure 9 shows the entity relationship diagram of the Fix Localization System project.

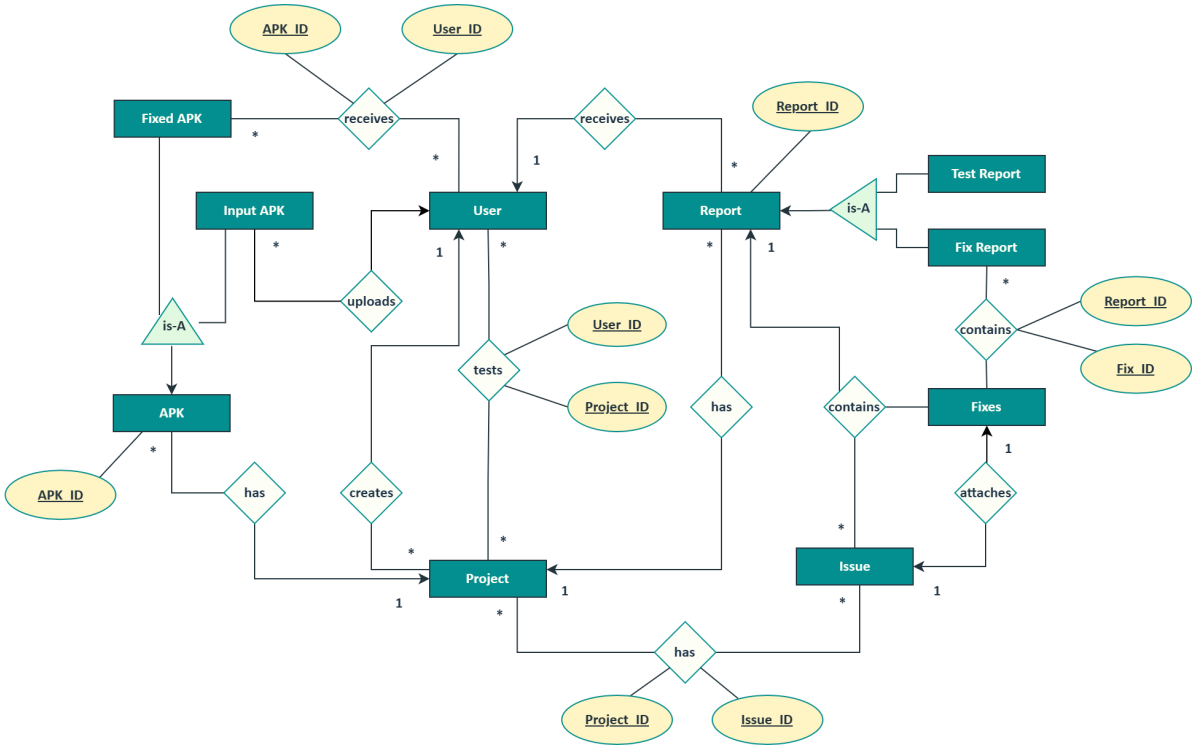

Figure 9: ER-Diagram of FLoS

2.4.4 Schema Tables

We have derived the following table from the ER diagram (Figure 9):

Data Object	Attribute	Type	Size
User	-User_ID -Username -Email -Password -Project	Varchar Varchar Varchar Varchar Object	16 40 40 40
Project	-Project_ID -Project_Name -Language -APK -Report	Varchar Varchar Varchar Object Object	16 40 40
Input APK	-APK_ID -Path	Varchar Varchar	16 40
Fixed APK	-APK_ID -Path -Version	Varchar Varchar Varchar	16 40 40
Issues	-Issue_ID -Title	Varchar Varchar	16 40
Fixes	-Fix_ID -Description	Varchar Varchar	16 40
Report	-Report_ID -Version -Project -Status -Path	Varchar Number Varchar Varchar Varchar	16 (8,4) 40 40 40
Test Report	-Issues	Object	
Fix Report	-Fixes -Comment	Object Varchar	 40
Has	-Project_ID -Issue_ID	Varchar Varchar	16 16
Tests	-Project_ID -Tester_ID	Varchar Varchar	16 16
Receives	-APK_ID -User_ID	Varchar Varchar	16 16
Contains	-Report_ID -Fix_ID	Varchar Varchar	16 16

3 Architectural Design

Our web application will follow 3-tier architecture - Presentation Layer, Logic Layer & Persistence Layer.

Figure 10: 3-tier architecture of FLoS: Fix Localization System

The presentation layer manages user interaction and communicates with the logic layer. The logic layer consists of APIs to complete communication with the presentation layer. It also processes the input and the bridge for the output to the client. The persistence layer stores and retrieves data from the system.

For FLoS, we constructed some system level structural organization, called architectural patterns. Here, we used the Model-View-Controller (MVC) pattern as the primary pattern for front-end architecture. The User Model is a representation of application data, meanwhile the view component is where the user interaction takes place. The controller manages interactions between model and view.

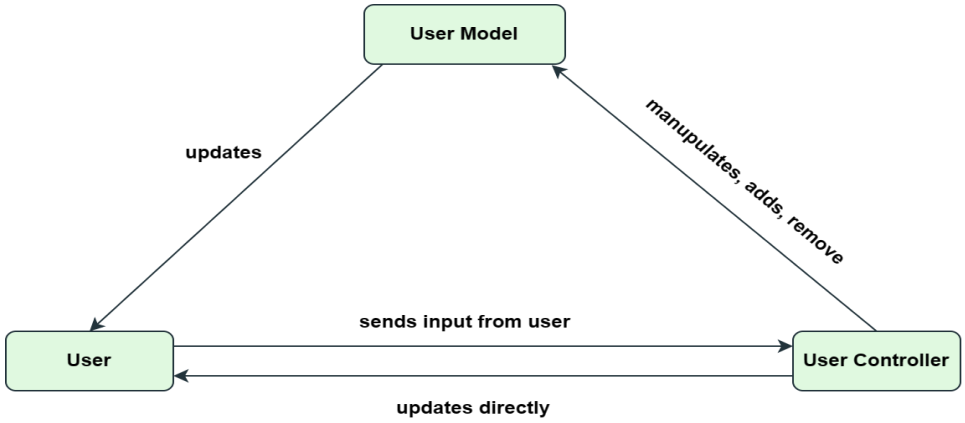

Figure 11: MVC pattern of User Component

Other than that, we used the pipe-filter pattern in the back-end. Localization processes will be broken down into multiple steps to find defects and then fixes for every defect, meaning a different outcome every time. For that purpose, we integrated the pipe-filter architectural pattern in our solution design. There are different filters which consume and transform data. Pipes are the connectors to pass it to another channel. Here, we used scanner, parser, marker etc. as filters to transform data streams and generate outputs.

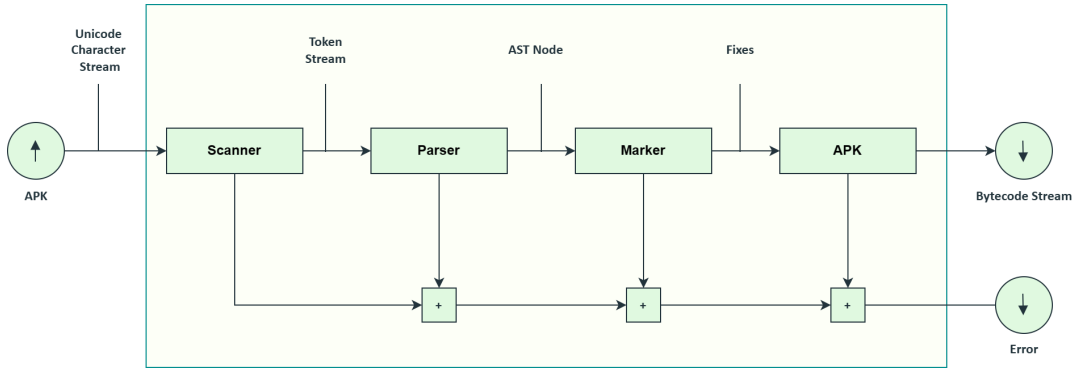

Figure 12: Pipe-filter pattern for issues & fixes

3.1 Choices & Trade Offs

- **Handling localization issues**: Handling localization issues in text while keeping the GUI consistent is a very time consuming task because we have to compare the generated fixed GUI with the original GUI again & again for fixing issues. So we choose to let the user select the localization issues he wishes to resolve. It decreases the solution space so we can better converge to solutions with less time.
- **Managing translations**: It is very difficult to provide proper localized translation due to local dialects. So we generate a default translation file that the user may edit to accommodate better translation. Our tool no longer remains totally automated but it does lead to a better solution.

4 Management Plan

4.1 Team Member

Serial No	Member Name	Role & Responsibility	Area of Expertise
1.	Shafiq-us Saleheen	Project Coordinator - Project Coordination & quality assurance	Project Management Software testing
2.	Hasnain Iqbal Shirsho	Back End Developer - Developing back-end & system integration	Machine Learning Back-End Development
3.	Moshin Ibna Amin	Designer - Identifying architectural patterns & component analysis	Android Development Software Architecture
4.	Shartaz Sajid Nahid	Front End Developer - Designing UI & implementation	Software Architecture Front-End development
5.	Abhijit Paul	Researcher - Researching about the topic & deriving the solution	Research activity Requirement Analysis
6.	Rupali Tasnim Samad	Requirement Analyst - Documentation & schema design	Software Requirement Analysis System Design

4.2 Team Coordination

For team management, we followed an incremental process model. We chose it because requirements were reasonably well-defined for our system. Additionally, there was a compelling need to provide a limited set of software functionality quickly and then refine and expand on that functionality in later software releases. The roles and responsibilities are chosen according to the expertise of the member. In each increment for communication we all had meetings in Google Meet. For planning and modeling, Moshin, Rupali & Abhijit did the requirement analysis, designing the solution and documentation. For the construction part, Hasnain & Shartaj did the backend and front end coding. Our project coordinator Shafiq did the testing and quality assurance and coordinated in every step. For working together on documents, we used Google Docs. We have used Google Drive for assembling all our work documents. To make diagrams, we used Draw.io. We used Trello for task management.

5 Source Code Management

To work collaboratively, we used Git and our project is hosted at GitHub[1]. Each of us published our code to the remote GitHub repository to keep everyone updated.

6 Time Management

Here is a timeline of how we completed the whole project using incremental process flow -

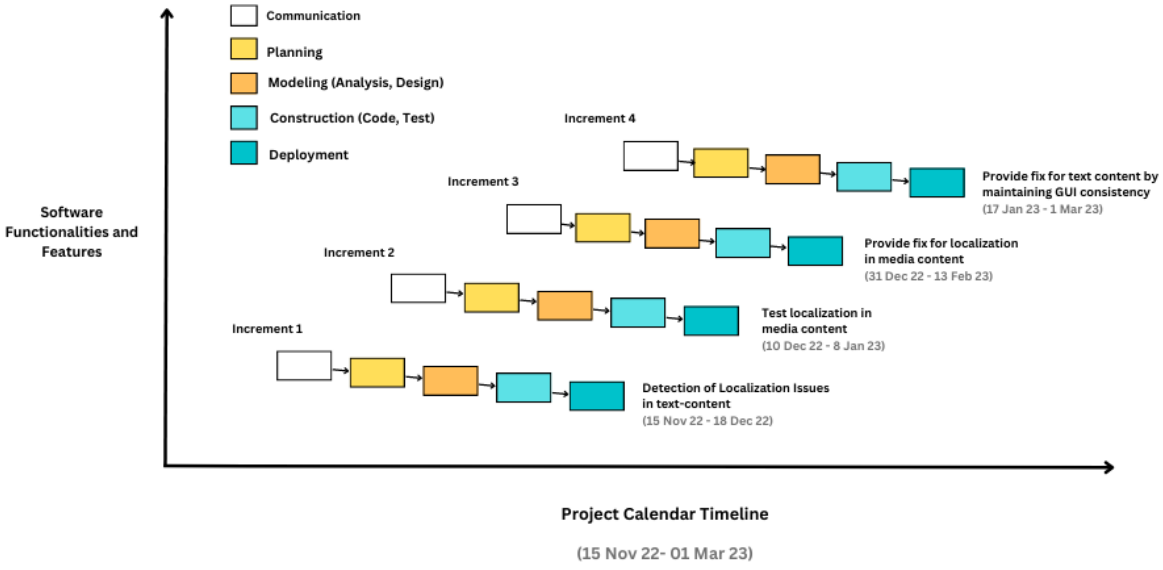

Figure 13: Time management chart for incremental process flow model

[1] https://github.com/saleheenshafiq9/FLoS-FixLocalizationSystem

7 Implementation

For our tool, we are going to use the following frameworks and technologies - React JS as front-end framework, Django as back-end framework and PostgreSQL as database, MinIO as object storage.

Our implementation consists of two segments - Detection & Fix. To fix localization issues, we had to generate text-based translation and handle GUI issues.

7.1 Localization Issue Detection

To detect localization issues in an android apk, we analyzed apk files and extracted xml files. So -
1. Insert APK - The user uploads apk as our input. The apk is stored in the object database. The user also selects the languages in which he wants to localize the app.
2. Construct AST - We then generate AST(Abstract Syntax Tree) from source code to navigate & extract information from code. We used *'py_ast*' , a python library to extract the AST from source code. We looked for image, audio or video references for the presence of any media contents. Here -

I. For image contents, we used the *Tesseract OCR* library available in python to detect and extract texts from images.
II. As for audio files, we used the required libraries such as *SpeechRecognition* and *pydub*.
III. For video subtitles, we used the *pycaption* library to parse subtitles.

3. Identify Markers - We traverse the AST and look for hard-coded specific patterns (string literals, constant variables, comment markers etc.) that indicates whether a string is being used for localization.
4. Detect Language - Now, we use the strings from identified markers and apply the *langdetect* library in Python to detect the language of the strings.
5. List Localization Issues - For the selected language, if the detected language of the text is of different language, then it is identified as a localization issue.

7.2 Fix Localization Issues (Text & Audio)

To localize the app, we created additional *strings.xml* files for the language and region that the user has selected. The texts will be translated using - MyMemory API (because it's free) and presented to the user as a translation file. We will translate identified captions, speech and subtitles into the desired language. For audio files, we will use a text-to-speech library, called *pyttsx3* to generate new audios. If the user does not like a translation, he will be able to edit and add his own translation in the translation file.

7.3 Fix Localization Issues (GUI)

After translation, the length of texts may become bigger that may not fit in the previous GUI element. So we need to increase or decrease the properties of the elements. We will do so in the following manner to keep the GUI consistent.

1. Build Visual Hierarchy Graph - This stage converts the XML file of apk source code into a hierarchy graph. Here, nodes are all the elements (button, text area etc) of the activity (the window in which the app draws its UI.) having an element id.

2. Cluster Elements - Each node from the Visual Hierarchy Graph is clustered using their XPath, size & class properties. We use the DB-SCAN[2] clustering algorithm because this particular technique is well suited for our problem since the algorithm does not require predefining the number of clusters, and produces mutually exclusive clusters (i.e., hard clustering). Elements in the same cluster are visually related.

3. Build Size Relationship Graph - To keep the GUI consistent, we need to propagate the change of one element to another visually related element. For that, we construct the size relationship graph. Here, nodes are connected using a consistent edge if they are in the same cluster. Nodes are connected using a dependency edge if their property (height, width, spacing) includes keywords like “Fit Width”. The weight of the nodes are the ratio of height and width of the independent element.

4. Identify Problematic Elements - Elements with localization Issues are identified in the size relationship graph by comparing the required size of the translated text. If the required size is significantly greater or smaller than element size, then it needs to be resized and is identified as a problematic element.

5. Generate Subgraph - We generate a subgraph that is a transitive closure of the problematic elements.

6. Convert Subgraph Information into Genes - We used a genome sequencing algorithm because it can work efficiently in a large solution space. To run a genome sequencing algorithm, we need to convert the nodes of the subgraph into genes. Each node contains element ID, properties (height, width, spacing) & values and is used to construct a gene. Value (how much the property should increase or decrease) is initialized by genome sequencing algorithm and optimized after each iteration.

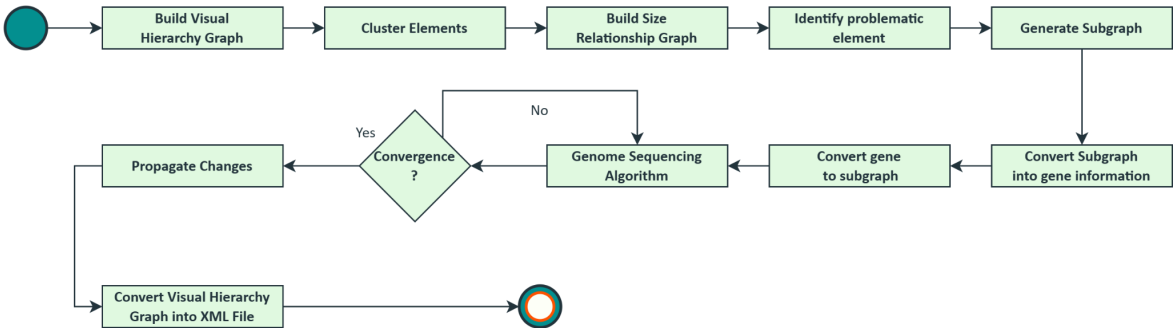

Figure 14: Fix Localisation Issues(GUI)

8. Propagate Change - After genome sequencing algorithm execution is completed, the values of each gene are reflected in the Size Relationship Graph by increasing or decreasing the property. Then this change is propagated following the edges.

9. Generate XML - XML file is generated from the Size Relationship Graph by converting it into Visual Hierarchy Graph and then, Visual Hierarchy Graph is converted into an XML file. This is the GUI fix for the apk file.

8 User Interface

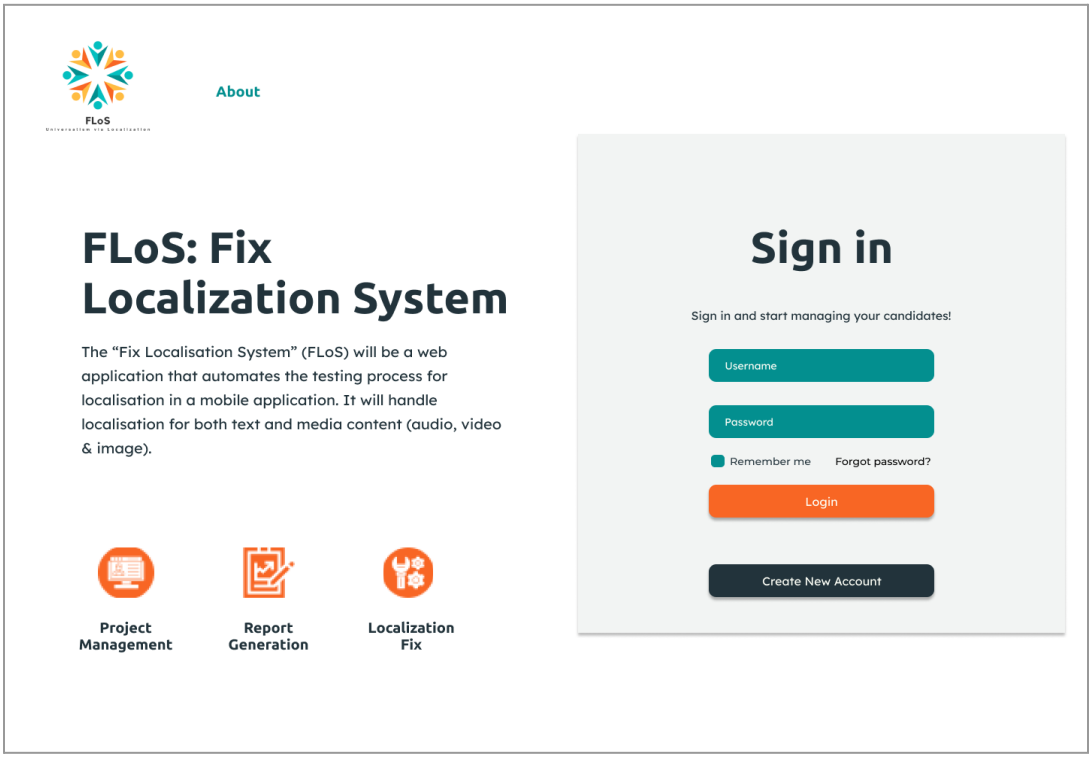

Figure 15: Screenshot of the sign-in page

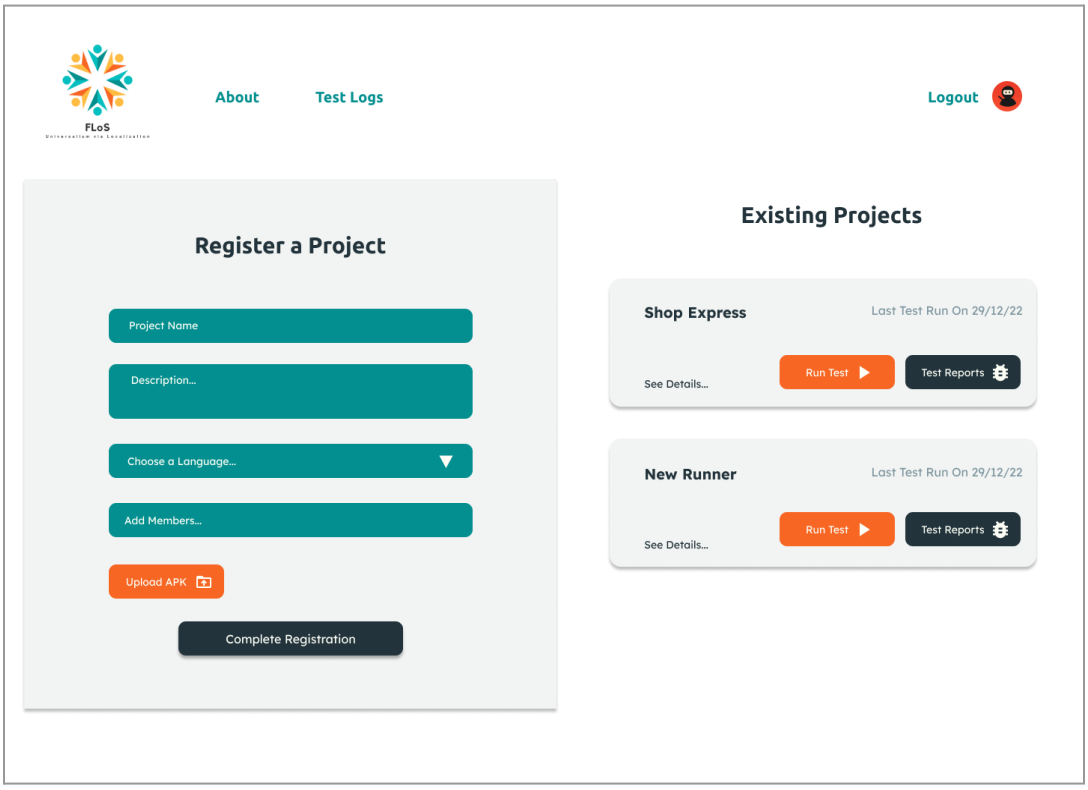

Figure 16: Screenshot of register project page

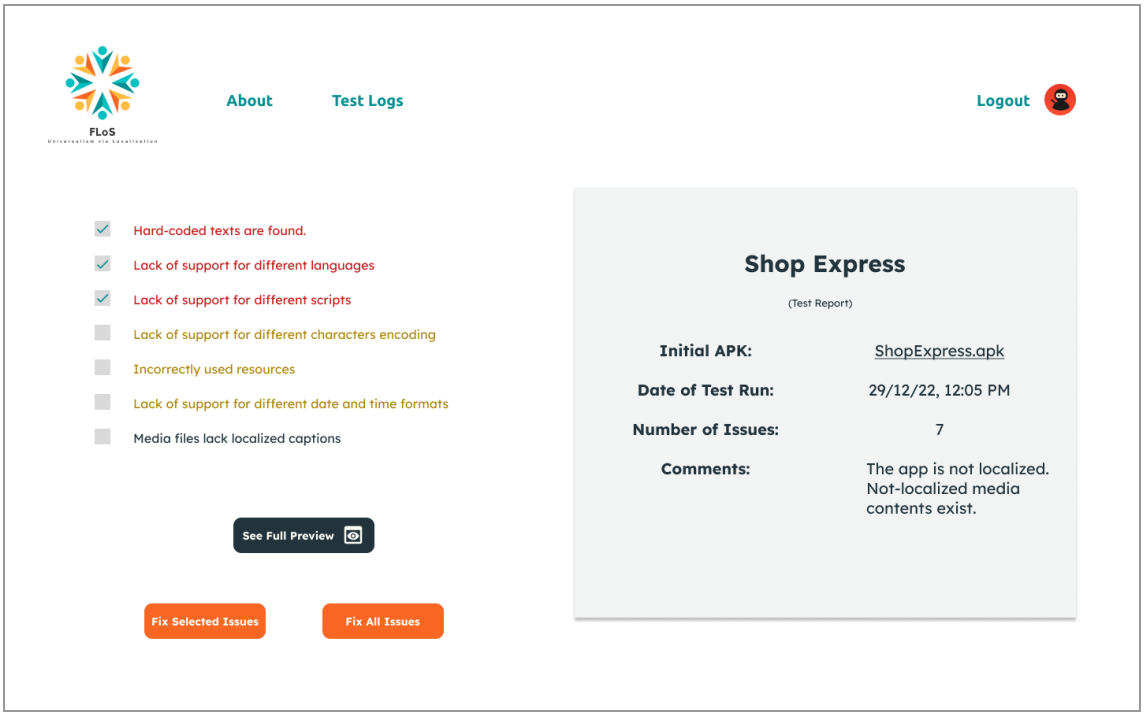

Figure 17: Screenshot of the issues detection page after test run

9 Verification & Validation Activities

Verification checks whether we are building the product right where validation checks whether we are building the right product. We planned these by considering the schedule and complexity of the system.

1. The documents related to requirements, design, test cases were thoroughly reviewed to find inconsistencies, logical errors, incompleteness and any other faulty detail.
2. We have planned to go through - unit, integration & system testing. For white-box testing, we carefully investigated the input to lead them to independent paths to view their control flow for rigorous testing. We used *unittest* for python to generate unit test cases. We will use *Selenium IDE* to do integration testing.
3. We wrote test cases after exploratory testing. We followed standard format to write test cases, by including test id, module, steps to reproduce, input, expected and actual results, status etc.
4. We did black-box testing by analyzing boundary values and robustness of inputs and outputs. We used *Selenium IDE* for this.
5. We used a static analyzer called *PyLint* here. It is used for white-box testing to check syntax, variables, code structure etc.
6. We used *Postman* to do API testing and inspected response headers to find defects.
7. To validate load, response time and analyze results, we will use *Apache Jmeter* as well.
8. We used *Burp Suite* for security testing. It is a comprehensive platform for web application security testing.
9. We have further testing to do on usability issues, integration etc. and some regression testing is required after completion of all modules.

10 Challenges & Lessons Learned

10.1 Challenges

The project was full of challenges. Some of the challenges were -

- Defining the scope of the project was difficult because Schwartz model [3] includes many human values of Universalism (e.g. accessibility, equality, world of beauty, social justice etc) and working with all of it was not possible within the time limit. We talked to the sponsors and it helped us to understand that our scope for the project is localization of language to ensure accessibility.

- Next challenge was finding a way to fix localization issues while keeping the GUI consistent. We found an approach [2] for it that clusters GUI elements to find the visually related elements and change them together to keep the GUI consistent.

- Producing translations of high quality was difficult to achieve, especially for idiomatic expressions and cultural references. So we allowed the user to edit the system generated translation file to accommodate the localized translation using jargons, dialects etc.

10.2 Lessons Learned

Throughout the project we have come to learn many things and some of the lessons we learned are -

- **Software development best practices:** We followed software development best practices such as testing, documentation from the beginning to the end for the very first time. We went through every phase of SDLC and learnt how it's like to develop following standard approaches.

- **Technical Skills:** By generating an AST of the app's source code, we learnt how to use libraries and tools to parse and analyze code, and how to navigate and manipulate the AST to detect issues in the code. We learnt how to use translation APIs (Mymemory api) to translate string resources in different languages & how to handle apk files, extract them, and repack them after making the necessary changes.

- **Software Testing Tools:** We learnt testing methods such as unit testing, integration testing, and performance testing. We learnt to use Apache Jmeter, Selenium, Burp suite, postman etc.

- **Research Knowledge:** We were inexperienced regarding literature reviews. But for gaining domain knowledge & finding a solution for GUI consistency issues we had to go through many research papers which will help us a lot in future.

- **Human Values:** We understood how vital it is to respect human values when developing softwares because it ensures that the software is designed and implemented in a way that is ethical, fair, and respects the rights and dignity of users.

11 Conclusion

We are very happy working with this project which is a new learning experience for us. At our core, we realized that we are the builders of the digital world and we have the opportunity to make it a beautiful place that respects human values, eliminates discrimination and gives everybody an equal opportunity (e.g. through accessibility). This philosophy will greatly aid us in our future works, in life.
Before we started writing the code we thoroughly went through the design phases. Our design phases are reflected in the requirements design and architecture design.

Before we started writing code we thoroughly went through the design phases. Our design phases are reflected in the requirements design and architecture design. There is still much room for expansion of this project -

- We will work on classification localization based on their severity.
- Creating detection tools and fix for other values of universalism such as - equality, accessibility, social justice, world of beauty etc.
- We will handle more corner cases for keeping the GUI constant. Such as handling padding & margin.
- We will cover the cases of dialects and regional variations while translation.
- We will fix Right-to-Left issues for translated languages (Arabic, Hibrew etc.) which will allow us to work with these languages more effectively.
- Our application will be able to test websites and provide localization fixes.
- We have rigorously studied discrimination as a core issue when trying to establish human values of universalism. We will work on detecting discrimination in a system.

We would like to thank Humphrey Obei and Professor Xiaoning DU for helping us from time to time by answering our queries.

References

[1] Ali S Alotaibi, Paul T Chiou, and William GJ Halfond. 2021. “Automated Repair of Size-Based Inaccessibility Issues in Mobile Applications.” In the proceedings the *36th International Conference on Automated Software Engineering (ASE)*. IEEE, pp. 730–742.
[2] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In KDD, Vol. 96. pp. 226–231.
[3] Shalom H Schwartz. 2012. “An overview of the Schwartz theory of basic values.” *Online readings in Psychology and Culture 2*, 1, pp. 2307–0919.