The first component is the Blockly library [23], on top of which accessibility is added. The keyboard module (input) provides keys to simulate the keyboard. The screen reader module (output) uses text-to-speech to output the code. The accessible block-based programming framework enables individuals with disabilities to code using a visual interface.

To design our solution, we rely on supervised learning techniques to effectively enable accessible block-based programming. We use machine learning algorithms to classify user reviews, into what type of accessibility guidelines they are addressing. We select six classification models for performance evaluation, namely Logistic Regression, Support Vector Machine, Gaussian Naive Bayes, Decision Trees, Random Forest, and AdaBoost.

To address the above-mentioned challenges, the goal of this paper is to help developers quickly distinguish accessibility-related problems and address them on a timely manner. We use an approach that involves data collection, data preprocessing, feature engineering, model selection, and model evaluation.

Step 1 - Data Collection: We collect accessibility-related datasets.
Step 2 - Data Preprocessing: We tokenize the data, perform stop-word removal, and use Lemmatization.
Step 3 - Sentiment Analysis: We classify the sentiment of the user reviews.
Step 4 - Feature Engineering: We use TF-IDF and BoW techniques on preprocessed review text.
Step 5 - Model Selection: We use six classification models for performance evaluation.
Step 6 - Model Evaluation: We validate the performance of the model, using evaluation parameters such as accuracy, recall, and F1-score.