**Malaria Screener**

Task-2 of Mboalab

**Task 2: Design of an app for the project based on the malaria Screener model**

Create a technical document with detailed list of tasks, description, tools needed to develop a similar app

**Abstract**

**Background**: In the field, light microscopy is frequently employed to diagnose malaria. It is, however, time demanding, and the quality of the data is highly dependent on the skill of the microscopists. Malaria light microscopy automation is a promising option, although it is still a challenge and a research topic in progress. Current tools are frequently expensive and need complex hardware components, making them difficult to deploy in resource-constrained places.

**Results**: We created Malaria Screener, an Android mobile application that makes smartphones a cheap and effective tool for automated malaria light microscopy. The mobile software scans thin and thick blood smear photos for P. falciparum parasites using high-resolution cameras and the computer capabilities of current smartphones. Malaria Screener's slide screening procedure includes picture collecting, smear image analysis, and result display, as well as a database for simple access to the collected data.

**Conclusion**: Malaria Screener speeds up, improves consistency, and reduces reliance on human expertise in the screening process. The software is modular, allowing other research groups to add their image processing and machine learning methods and models while collecting and analysing data.

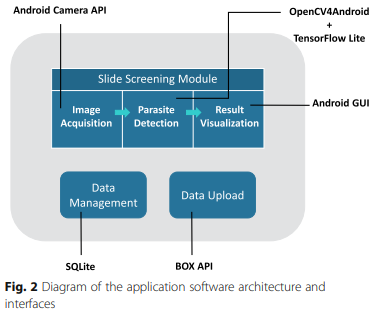
**Steps to make a similar App are as follows-**

1. **Software architecture for implementation**

Malaria Screener was created using object-oriented ideas. Figure 2 depicts a diagram of the application's architecture. A slide screening module, a data management module, and a data upload module are among the three independent modules. The slide screening module, which is at the heart of the system, is divided into three sub-modules that execute picture acquisition, parasite detection, and result visualisation in that order. The data management module saves photos and metadata collected during slide screening sessions, allowing users to access slides that have already been examined. Finally, the data upload module uploads the local data to an internet repository for archival purposes and system training.

Internal Tools Used-

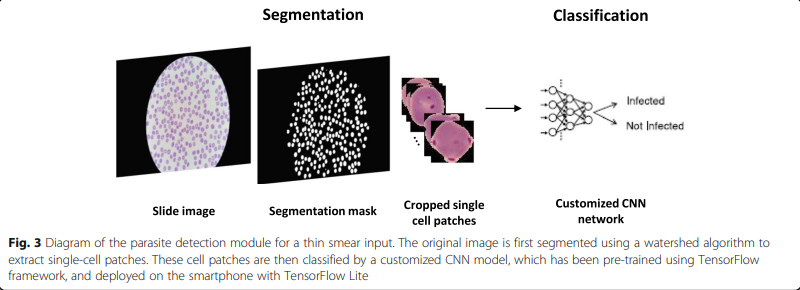
The front-end user interface (UI) was built using Android API, while the back-end was built using a combination of libraries such as OpenCV4Android (opencv.org/android/), TensorFlow Lite, SQLite, and Box API.



1. **Module for screening slides**

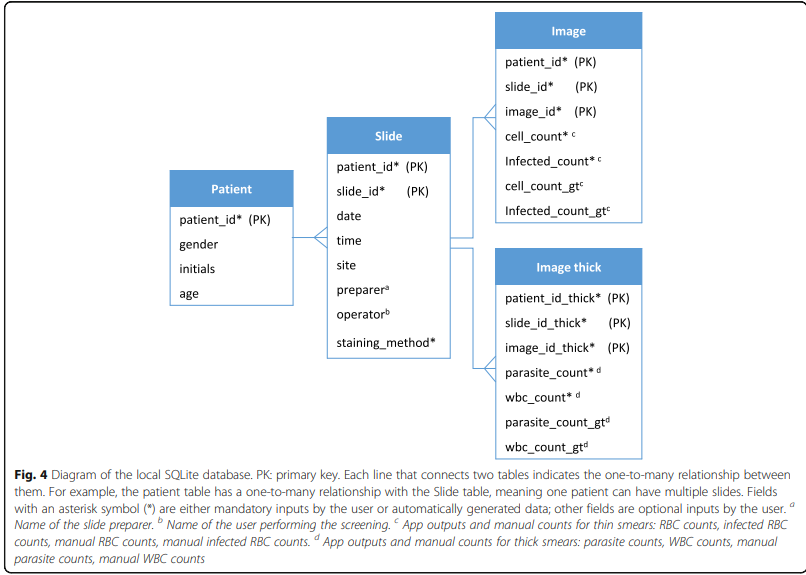
As previously stated, three distinct sub-modules screen a slide for malaria parasites in a sequential manner. This pipeline's first module is the image acquisition module. We used the **Android Camera API** to create a custom camera method for this module. This comprises a Camera object that controls the camera's inherent properties and a **Camera Preview** object that shows the user the preview image. When an appropriate field of vision becomes visible during a screening session, the user presses a button to capture the image. The Camera object asks the greatest resolution offered by the **smartphone camera** and stores the taken image as PNG, a lossless compression format, to achieve the best image quality possible. The parasite detection module receives the collected image as an input. Malaria Screener can look at thin and thick smears for P. falciparum infections. The image processing in this module is shown in Figure 3. The goal is to count the number of contaminated red blood cells (RBCs) and the overall number of RBCs in a single thin smear picture. On the other hand, the purpose of a thick smear image is to count parasites and white blood cells (WBCs). To address the two circumstances, the parasite detection module comprises a ThinSmearProcessor class and a ThickSmearProcessor class. A thin smear image is initially segmented to detect RBCs with ThinSmearProcessor. The original image has been cropped to remove small RBC cell patches. Parasite candidate patches that cover the typical size of a parasite are cut from a thick smear image with thickSmearProcessor. In the event of a thin smear, pre-trained **Convolutional Neural Network (CNN)** models are used to produce binary classifications: infected versus uninfected RBC, or parasite vs background in the case of a thick smear. **TensorFlow** and **Keras** are used to train the **CNN models** on a PC, and the trained models are saved in **HDF5 (.h5) format**. Following that, the models are transformed to **Protocol Buffers (.pb) format** and **TensorFlow Lite** is used to deploy them to the app.

The result visualisation module makes use of Result Displayer Activity, a user interface class that displays the detection results to the user. This class creates a down-sampled image with labels drawn on the infected RBCs (parasites for thick smear images). In addition, the numerical results are displayed in a table. These two outputs together show the user the computational outcome of the input smear.



1. **Module for data management**

Images and metadata are stored locally on the phone by the app. Images are saved in a specific folder on the **internal hard drive**. Images from the same screening session are organised in subfolders inside this folder. The photos' metadata is saved in a local **SQLite** database. A patient table, a slide table, a thin smear image table, and a thick smear image table are all included in the database. Figure 4 depicts the database's structure in greater detail. The data management module also includes a graphical user interface (GUI) that allows the user to view the photographs and metadata stored in the SQLite database.



1. **Module for uploading data**

The database's images and metadata can be exported and posted to an internet repository. The uploaded data can be used to evaluate the app's performance and, with more training, to improve the parasite detection module's classifier.

An upload event can be started one of two ways. The first approach is to use the database UI to initiate an upload event. This option causes the app to scan for any data that hasn't been uploaded yet and then upload it.

However, because there could be many terabytes of photographs to upload, this type of bulk upload can be a highly time-consuming task. As a result, we included a new upload option in which the app tries to initiate an upload event after each screening session. This event will attempt to transfer the data from the current session as long as a Wi-Fi connection can be identified.

This module's backend is built using a combination of **Android** and **Box API**. The upload method is implemented using the **Android Service** class and the **Thread class**, which runs the upload operations in a separate thread. It enables the user to do other things while the upload duties are being completed in the background. The Box API is used to create functions for uploading files to a **Box repository**.

1. **Results**

As a lightweight alternative to automated malaria light microscopy, a quick and effective mobile app is being created. This section explains how the system works during a slide screening session.

(1) After mounting the smartphone on top of the microscope with an adapter, the user can begin a session from the app's main screen.

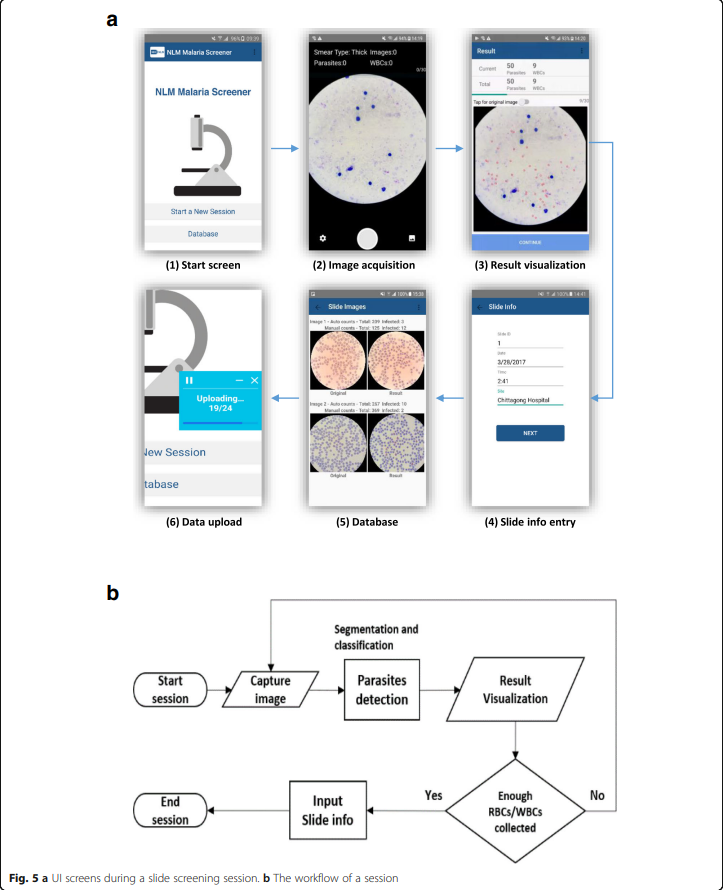
(2) At the start of a session, the user is presented with a camera preview screen, and at this point, the user can utilise the button at the top of the screen to select the smear type (thin or thick). The user can then look for an appropriate field of vision on the slide and record the image by pressing the camera button.

(3) The software will then begin processing the acquired image, and a result visualisation page will appear on the screen, displaying the detection findings. In thin smear mode, for example, the number of infected and total RBCs, as well as a running total, are displayed. In addition, the app displays a result image with contaminated RBCs highlighted in red. While the user captures more smear photos, steps (2) and (3) are repeated. When the total number of RBCs hits a user-defined maximum, the loop ends.

(4) After that, the software takes the user through a series of displays that ask for information about the slide, such as the slide ID, staining procedure, and hematocrit value.

(5) The session then comes to a close. Both the photographs and the metadata are saved locally and can be accessed at a later time through the database UI.

(6) Finally, the app sends the saved data to the central Box repository by triggering an upload event. In the meantime, a hovering widget hovers over the app screen, displaying the upload status.



1. **Testing**

The algorithms we created for the slide screening module were put to the test. We obtained and annotated Giemsa-stained thick and thin blood smear pictures from P. falciparum-infected patients, as well as 50 healthy people. We used five-fold cross validation to evaluate the performance of our system for thick smear, and we used five-fold cross validation for thin smear at both the patch and patient level. On a patch level, we compared our outcomes to the state-of-the-art.

**Conclusions**

We present a low-cost, quick-to-use smartphone app for malaria screening. We show how the app can (a) automatically screen slides for P. falciparum malaria and count infected red blood cells and parasites in thin and thick smear images, and (b) manage the photos and metadata generated during the screening process, which can be utilised to improve the image analysis model.

**Requirements**

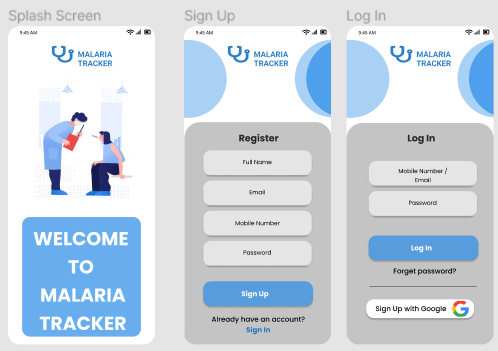
**Operating System**: Android.

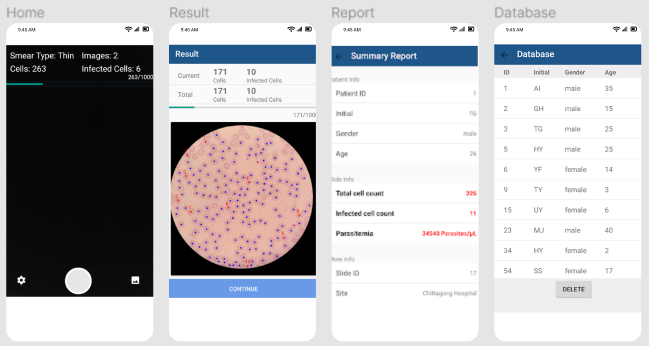
**Programming language**: Java, C++ (for Android Native development).

**Other requirements**: Android Lollipop/5.0 and above.

**License**: Open Source Software.

**Similar App Malaria Tracker Designed By Me**

****

****

**BY-**

**Radhika Saini**