OELP Final Presentation

Software for Microscope

Faculty
Dr. Albert Sunny

VM Sreeram - 112001051
Nideesh N - 112001028

Description of the problem statement

Build a robust production-grade software for a low-cost automated fluorescence microscope.

Approach taken

The software is an Electron app, can be installed on Ubuntu Core and displayed via Ubuntu Frame. HTTP server simulates diagnostics during bootup, MongoDB stores user details, and OAuth verifies admin registration. Express NodeJS server simulates microscope arm movement. Limited WiFi interface. Reports saved as CSV files, viewable in the app or via USB. Credits required for tests and maintained locally. UML sequence diagram shows communication flow between user, device, and servers.

Technologies used

Sequence Diagram

A sequence diagram shows process interaction arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality

Link to our sequence diagram - Link

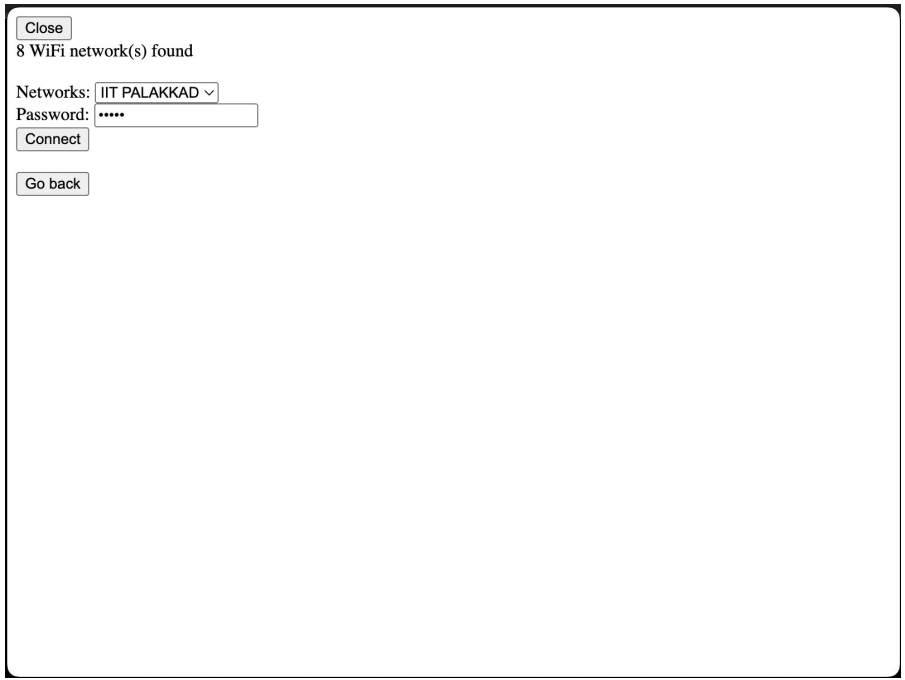
Functionalities of the application

- Check if the API Key is present in the device.
- One time verification of admin using OAuth, fetching API key from company DB.
- Set up Admin username and password.
- Limited interface for WiFi connectivity.
- Simulation of hardware diagnostics.
- Periodic checking of network connectivity.
- Admin can register users with privilege level.
- Registered users can perform tests.
- Logging of actions done by users.
- Simulation of microscope arm and minimal report generation.
- Reading and displaying CSV files and saving them to USB.
- Tests consume credits and may require WiFi connection.

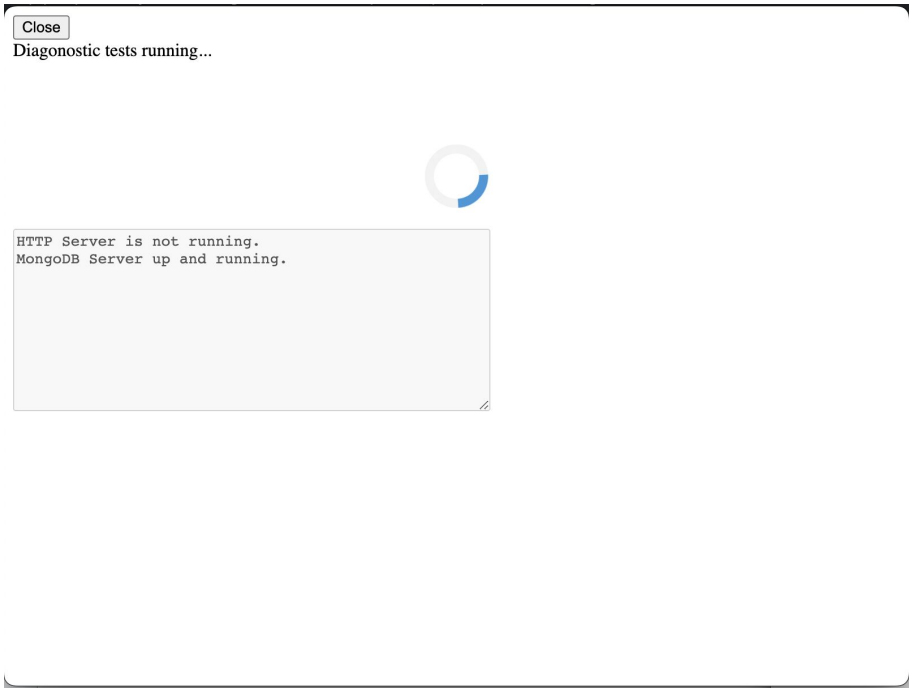
Project Structure

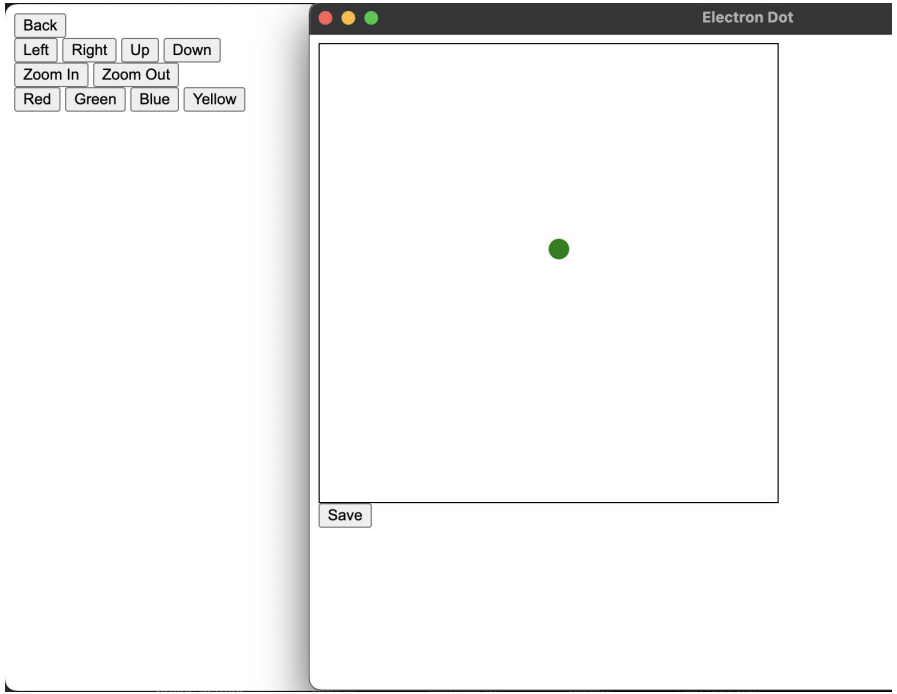
Robust


Close
ConnectWiFi
Hello
Username:
username
Password:
password
Submit


Close
8 WiFi network(s) found
Networks:
IIT PALAKKAD
Password:
Connect
Go back

Simulations





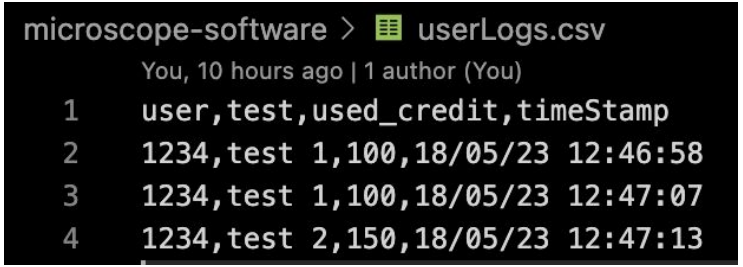
CSV Files

CSV Reader

Click the button to open a CSV file:

Open CSV | Save to USB


```
{"x":"230","y":"190","w":"14","c":"red"}
{"x":"230","y":"190","w":"14","c":"red"}
{"x":"230","y":"190","w":"14","c":"green"}
{"x":"230","y":"190","w":"14","c":"green"}
{"x":"230","y":"190","w":"14","c":"blue"}
{"x":"230","y":"190","w":"14","c":"yellow"}
{"x":"230","y":"190","w":"14","c":"blue"}
{"x":"240","y":"190","w":"14","c":"blue"}
{"x":"250","y":"190","w":"14","c":"blue"}
{"x":"250","y":"180","w":"14","c":"blue"}
```

Further steps

- Implementation of an online credit system using bank APIs and company databases.
- Saving states in the file system and recovery from last saved states in case of failure.
- Making the snap of the whole application and publishing in the snap store. Since MongoDB is server based, making a snap of it is not possible so we have to find an alternative (file system based) to store info such as registered users and credits.
- Add documentation for the application.
- Implementing actual hardware APIs instead of simulating them and proper error handling for hardware failure.