Web App for Microscopic Cell Analysis*

Sahu, Samyak[1]

Birla Institute of Technology and Science, Pilani, India
f20180421@pilani.bits-pilani.ac.in

Abstract. We have developed a open-source, Flask-based web application that can be used for cell classification for medical analysis. The app takes in the microscopic feed of a urine sample, in an image or video format. In case of a video format, an additional layer of sharp-frame detection is also done. After that, it identifies the number of type of cells. Based on their number and relative proportion, it detects what renal disease could the sample indicate. The application is expected to aid untrained staff at hospitals and infirmaries in remote villages, where there's a lack of expertise in interpreting and classifying various types of cells. The application could further find its way to help doctors automate, at least a part of, their diagnostic process – not just for urine samples, but also for detection of other cell samples suspended in a liquid medium.

Keywords: Flask · Web Development · Deep Learning · User Experience · Accessibility · Biomedical Image Analysis

1 Introduction

The Problem Medical image analysis plays an important role in the field of diagnosis. From X-rays to MRI scans, there are a lot of ways in which we can examine the body from the inside and see if anything is out of order. But it is often not required to use invasive strategies to detect anomalies in the human body. Sometimes we can extract a sample of a bodily fluid, such as blood, urine, sweat, saliva, mucous, semen, etc., and can predict from the portion of a whole, the presence of any suspicious cells or any type of cells in an abnormally large proportion. This type of diagnosis has always been more prevalent in the medical field, primarily due to its non-invasiveness, ease of conduction, cost and safety factors.

For higher level analyses, there's often a need of trained professional staff. In the Indian context, especially in remote villages and towns, it is found out that there is a disproportionate level of false positives and false negatives coming up due to *(a)* human error during sample examination and *(b)* miscarriage of samples, which can lead to their contamination. Both the factors can be, in part, attributed to the lack of trained staff in hospitals and infirmaries of remote villages. In addition, there's often a problem of a lack of robust infrastructure due to economic problems.

* Supported by Medical Center, BITS Pilani.

To counter the problem with (a), as mentioned above, we have attempted to leverage the power of Deep Learning-based image classification algorithms to make better predictions on the type of cells and their proportions. We attempted to build a software application around such a classification technique, that we attempt to explain in this paper.

The Solution We built an cell classification software that is designed to be easily accessible to be used by hospital staff in remote villages and infirmaries. When trying to develop a software solution for such a demographic, we had to keep in mind the following factors:

(1) Easy operation This is important so that even technologically-challenged individuals can operate the application with ease.

(2) Eliminating the possibility of human error The central aim of this project has been to reduce human intervention and thus eliminate, at best, or reduce, at least, any instances of human error that lead to inaccurate diagnosis.

(3) Quick, conclusive results with a well-defined confidence interval It is naturally expected of automatic systems to reduce the effort or time taken of a certain process as much as possible. So its desirable that the results take minimal time to process and show up. The Deep Learning algorithm used also provides us with a confidence interval for experienced professionals to evaluate the predictions and make an informed decision thereafter.

(4) Fully operational even with patchy or low-bandwidth network connections A large numbers medium-to-small towns and villages still have limited access to electricity and internet. Intermittent power cuts are a norm and 4G internet is yet to find penetration in many areas, especially around the state of Rajasthan. There may also arise the issue of hardware constraints with individual users. Keeping this in mind, we had the additional constraint to design this feature-heavy application such that there's minimum computational dependence on the client side. The application is designed with low bandwidth requirement, with measures to do discussed soon after this section.

(5) Easy open-source support We wanted the project to be easily replicable and one that could be easily improved upon. We had to keep in mind the needs to the developers who could adopt project up, mould it and improve it according to their needs. A lot of access points and extensive documentation had to be provided so that developers could understand the underpinnings of the code.

Remark 1. Keeping the following challenges in mind, we started out with a thorough analysis on what could be the most suitable type of application software paradigm for our needs.

2 Application Software Paradigms

There are many kinds of software architectures/paradigms available to us at present. At the same time, new paradigms are slowly being developed by corporations and research labs to improve on existing designs and provide the best out of the pre-existing ways to build an application.

The following section attempts to analyze the type of application software paradigms that would be best suited for our needs, on the assumption that an increasing number of rural areas have access to mobile data-based internet, an image capturing device (a mobile camera), a microscope and a personal computer at one such infirmary.

2.1 Native Applications

Native applications are those types of application software which are developed to work on a specific platform. Its form and function are greatly dependent on the device-specific hardware and operating system.

The advantages and disadvantages of native apps are the following:

Advantages Native app structure offers many advantages such as,

Closeness to system The application's proximity to the system allows efficient use of system resources and maximum utilization of hardware.

Compatibility Issues around compatibility are little to none in case of native applications, because the apps are tailored-fit for the specific platform and the hardware used.

Disadvantages There are some disadvantages to the native app structure as well:

Dependence on native system The dependence on the system can come off as a disadvantage because the application's functionality is sometimes greatly inhibited by the hardware it is running on.

Inability to run cross-platform The lack of cross-platform support comes in the way of user experience, in an age where a single user often owns multiple devices having different form factors and operating systems.

Development-specific issues Finding platform-specific development environments and talent to develop for some obsolete system can be a highly challenging prospect, in time when cross-development platforms are taking the center stage.

2.2 Web Applications

To work on the shortcomings of native applications, the ubiquity of the Internet was exploited to make web-applications, which are application software running completely on the web.

Advantages The advantages of web applications are as follows:

Synchronizing capabilities The online hosting provides the app the liberty to be accessed from several platforms.

Ability to collaborate Multiple users can work on the same project and thus, effective collaboration among teams can be carried out.

Better development cycles There is a strong ecosystem of robust frameworks available for developing web applications (one such is used in our application). Updates can also be quickly rolled out, which ensures better iterative development lifecycles. Agile development practices can also be effectively carried out with these applications. [1]

Less installation dependency There is little to no need to install additional drivers/dependencies when using web apps. They are also highly *device agnostic*, that means they don't rely much on device hardware for their performance.

Disadvantages The disadvantages in case of fully web-based application models are:

Speed and compatibility issues Since the apps are more "distant" from the device as compared to native applications, they are not often optimized for speed for certain devices and operating systems. The users using obsolete ecosystems or outdated hardware can therefore find themselves at a disadvantage here.

Dependence on "being online" There is often the requirement of a steady internet connection for the web application to work properly. The use of asynchronous web technologies such as Ajax can limit the amount of data needed to be transferred, but an active, fast internet connection is more often than not, required.

2.3 Progressive Web Applications

Progressive Web Apps (or *PWAs*) were developed by Google. They represent a unique programming paradigm which ensures strong usability on both the server and the client side. (see Fig. 1) In essence, they are web apps which make extensive use of browser APIs to give a native-app like experience, including the capability to fetch system information, being installable and having partial to complete functionality even when the user goes offline. [2]

The advantages and disadvantages of PWAs are as follows:

Advantages PWAs are garnering a lot of support from the developer community due to their advantages.

Speed PWAs are often specially optimized for speed, even with moderately poor internet connections. The navigation of such apps is comparable to their offline, native counterparts.

Choice of installation This is a huge advantage as it offers a great flexibility for the user. The capability to install the app completely (or make use of some functionalities partially) offers unprecedented reliability.

Disadvantages There are some disadvantages associated with the PWA architecture as well

Demanding quality standards The build requirements of PWAs can come in the way of development (as they did in our project) for organizations looking for a basic level of viability. It therefore increases development time of the application. More often than not, web apps need to be converted to a PWA, and making a PWA from scratch isn't usually the path taken.

Limited Browser Support PWAs often require a Chromium-based browser to run on.

No dedicated marketplace This could be a business/availability aspect that can come in the way of distribution of these PWAs. Native apps often have a "store" from which applications for a certain system on a specific set of framework(s) can be downloaded.

New development community The paradigm has only recently been released which makes it hard to find developers and frameworks (other than those directly or indirectly endorsed by Google) which are instrumental to create these PWAs. The community around PWAs is not matured as of yet.

After thorough research, we decided to go with a web-app, with plans to go Single Page Application (SPA) model and then to a PWA model at a later stage.

Remark 2. Having chosen the most suitable app development paradigm for our application, we moved over to adopting the right framework to develop the application upon.

3 Choosing the Right Framework

A web framework (WF) or web application framework (WAF) is a software framework that is designed to support the development of web applications including web services, web resources, and web APIs. Web frameworks provide a standard way to build and deploy web applications on the World Wide Web. [4]

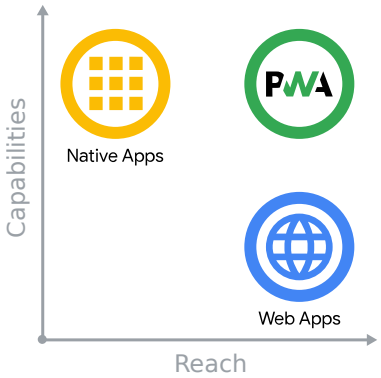

Fig. 1. Capabilities v/s reach of native apps, web apps, and progressive web apps.[3]

A framework was important for our development cycle as it provides a development template in which code elements have to be placed as per requirements. It eliminates the need to build fundamental elements required to build a web application from the ground up, and takes care of needs pertaining to structuring, deployment and debugging of the application.

Here's an in-depth analysis of the most prevalent web frameworks to develop a web application, followed by our decision based on our development experience, timeline of development and ease of integration with the image analysis scripts.

3.1 Angular

Angular is a TypeScript-based open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. [5] We decided to start off with Angular as the framework of choice for our project, as it the most widely used web framework used for making Progressive Web Apps.

In the process, we observed the following advantages and disadvantages.

Advantages Web development in Angular offers the following benefits:

Usage of TypeScript TypeScript, as opposed to JavaScript, is a typed language, which allows easier checks and debugging routines, in addition to the robustness and consistency in data usually seen in typed languages.

Robust MVC architecture With the Model-View-Controller architecture, it is possible to isolate the app logic from the UI layer and support separation of concerns. With Angular, the need to neatly organize the app into Model, View and Controller often becomes a necessity.

Modularity Angular is well-known in the developer community for the ability to import and export modules to support operation.

Disadvantages There were a lot of disadvantages associated with this framework as well, which led us to not use it in our project.

Tight development structure Even though it increases the modularity of code; the development architecture is not recommended for a beginner to intermediate front-end developer. The architecture can come in the way of some functionality.

Steep Learning Curve Slowed down the rate of development quite considerably. The onboarding tutorials of the framework are difficult to follow for developers lacking industry-level experience.

Large application size With all the dependencies in place, Angular apps weigh onwards of 200MB and contain a lot of convolutions in its file tree. The architecture starts failing for Machine-Learning based applications, where there already are a lot of dependencies such as model weights and other libraries.

3.2 Vue

Vue.js (pronounced as "view dot js") is an open-source model–view–viewmodel front end JavaScript framework for building user interfaces and single-page applications. [6]

Advantages Vue.js is useful in the following ways for frontend web development:

Ease of installation As opposed to the obligatory installation requirements for Angular (including a Command-Line Interface), the start can be as easy as importing a JavaScript file in an HTML template.

Lightweight architecture Frontend architectures developed over Vue.js can be very lightweight and can be effectively used for Single Page Application (SPA) development.

Virtual DOM manipulation Finding nodes that have been changed in the virual DOM is much faster. This way we bypass the bad performance of searching and updating multiple nodes in a large scale application.

Disadvantages

Relevance to the project After the models were built, it was realized that the backend integration needs a greater attention. The frontend did not require significant DOM manipulation (because we weren't building an SPA first)

Steep Learning Curve The learning curve was not as big as Angular, but going through the concepts of components, routing and creating state-server communication modules and directives took away a lot of development time from the neccesary backend development, and therefore, were prioritized out of the equation.

3.3 Django

Django is a Python-based free and open-source web framework that follows the model-template-views architectural pattern. It is maintained by the Django Software Foundation, an American independent organization established as a 501 non-profit. [7]

Advantages

Robustness Django is a full-fledged backend framework with a "batteries included" philosophy followed for maintaining a database migrations, sessions handling, etc. It also has excellent scalability, accentuating its versatility even more.

Pythonic support Django supports a host of packages and all code in Django is written in Python - which is a very user-centric, intuitive, general-purpose programming language. That means it can extensively be used in hosting machine learning models, create Pythonic UIs, in addition to setting up the backend of the application.

Disadvantages

Prerequisites An understanding of the entire system is required to proceed with working on Django, especially when short deadlines are considered.

Opinionated The framework is strongly opinionated, which gives it a monolithic feel. [8]

Bloated for small projects Django applications can prove to extremely heavy and bloated, especially for starters. Our use case also involved a lot of load in the form of machine learning models and dependencies, which was hard to accomodate in this framework.

3.4 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries to operate. [9]

Advantages

Lightweight structure Flask is a small library on its own which takes care of template rendering and some database features can be used with the help of pre-existing third-party Python libraries.

Flexibility Because there's no fixed structure, the Flask server architecture can is minimal and can be moulded to a project's requirements.

Pythonic support Like Django, the code in Flask is written in lucid Python and has a lot of community support, in addition to the hundreds of packages which can be made use of with this framework using the Pip package manager.

Disadvantages

Lack of robustness Things can end up being very convoluted in a Flask-based backend environment, because of Flask being a micro-framework with minimal functionality of its own. Therefore, it is not suited for applications that need extensive scaling.

Limited database-centric functionality It's not suited for enterprise-level applications with complex database requirements.

Remark 3. Considering our project requirements, only a backend framework was required. Flask was chosen as the backend framework of choice, to avoid language asymmetry with a JavaScript-based framework (a wrapper to interact with Pythonic machine-learning models would be used then, and it would have involved another layer of abstraction, potentially slowing the app further). We used simple HTML and CSS templates to avoid any additional bloats to our frontend.

4 Application Architecture & User Flow

We proceeded with a *Model-View-Controller* architecture, which is a prevalent application architecture in the software development industry.

4.1 Model-View-Controller Architecture

Model–view–controller (MVC) is a software design pattern commonly used for developing User interface that divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. [11]

The user flow was thought of and prototyped on *Figma*. Here's a flowchart visualization for the same. (see Fig. 3)

The app has a common introduction page, followed by two diversions for the *image upload* and *video upload* processes. Once the frame extraction is complete, the input processed so far (a sharp frame ready for cell classification) takes a common route once again. The frame is processed through for the presence of different types of cells and a classification with confidence interval is generated.

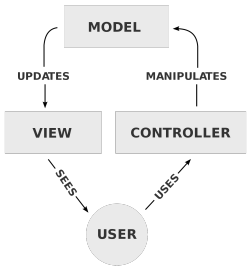

Fig. 2. Model-View-Controller Architecture - A diagrammatic representation. [11]

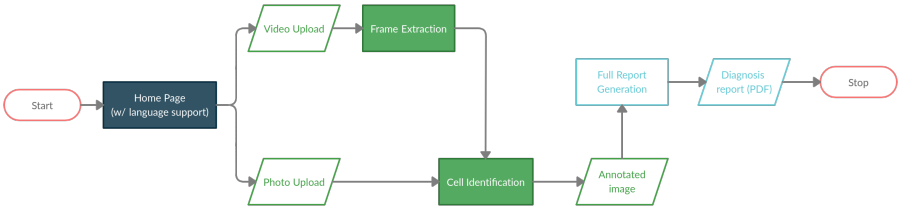

Fig. 3. A (proposed) flowchart for the user flow of the application.

5 Feature Description

5.1 Image extraction

The frame extraction process takes in a video output, possibly from a microscope feed, and extracts a sharp frame out of all the given frames in a video.

The algorithm used extracts frames from 'focusing-defocusing' videos of microscopic cultures and uses them to create a single sharp image of objects in the video. A multi-step focusing is often required because the cellular objects lie in different planes of focus.

5.2 Cell classification

The cell classification step involves a sharp frame as an input be it fed manually by the expertise of the user or extracted from a video by the sharp frame extraction algorithm as mentioned above.

A powerful image classification algorithm called YOLOv3 is being used for object detection and classification. It is a fully convolutional neural network. The model learns from the training images which are annotated and it predicts and classifies the cells on unseen images by denoting bounding boxes around the cells and labelling the class of cells. [10]

6 Accessibility Considerations

The application was specifically designed keeping a specific demographic in mind - the untrained staff in hospitals and infirmaries in and around the state of Rajasthan, as per the problem statement given by the Medical Center, BITS Pilani. We discussed the challenges associated with the same in section 1.2.

These are the practices we adopted to ensure a high degree of user accessibility

Language Selection A Hindi pipeline runs in parallel to the existing user flows. These are HTML pages rewritten in Hindi, so as to ensure virtually universal accessibility, given the state of Rajasthan has the second highest number of Hindi speakers in the country.

Pictoral indications A number of pictoral indicators have been put where ever neccessary, to ensure a smooth user experience. *Anchoring bias* was effectively utilized to help even the most technologically-challenged users reduce their cognitive load. [12]

Usage of character-based ideograms Character-based ideograms, popularly known as *emojis*, were used instead of rasterized (.png, .jpg, .bmp, etc.) or vector (.svg, .eps, .pdf, etc.) image formats.

This was done for two reasons in particular. First was, as discussed, *to facilitate the need for plenty of images as discussed in the point above.* Second, *was to reduce page load times.* A static image asset (rasterized or vector) uses anywhere between 200-500 kilobytes of storage. On the other hand, emojis are, in essence, part of an extended UTF-8 character family, which means a modern computer system interprets it as a character - occupying only 4 bytes in memory. This reduces the storage size of assets considerably (by a factor of 25000 to 62500) and this effect grows in prominence as the number of image assets increases in the application. This ensures faster browser load times and faster HTML rendering by the server.

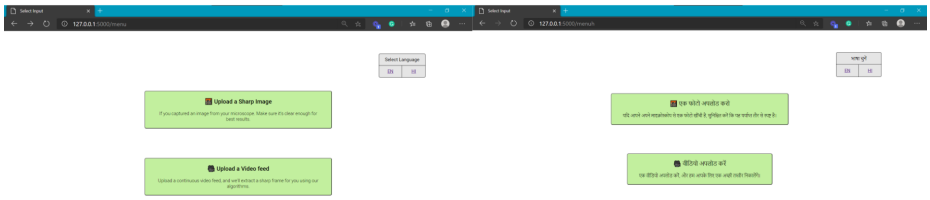

Fig. 4. A screenshot of the web app, displaying the iconography and language support

Usage of customized CSS files A UI framework such as *Bootstrap* or *Material* was earlier thought to be an strong candidate for displaying attractive UIs. However, due to the heavy nature of the suite, it was replaced by a single CSS file only containing natively-created classes. This further ensured small asset size, and hence, preventing instances of a bloated website which could reduce load times.

7 Future Scope and Implementation

Having completed with the first version of this application, we understand there is a lot of scope of improvement. There lies a vast array of applications imaging techniques like these could find.

We found that, during the course of this project and after it, that there can be several ways in which we can improve this application, such as:

Login functionality for hospitals We have thought of a login functionality, which will make it easy for each medical center to keep a record of its patients and retrieve their reports as and when required.

Full report generation There can be a provision of generating a full-fledged report (possibly in a PDF format) of the disease one could possibly have, along with the confidence intervals. This report could also contain additional information about the patient's health history, which doctors could use effectively to monitor their patients.

Conversion to a PWA There can be further development and the web app can be instead deployed as a fully featured Progressive Web App. For that, a JSON web manifest file will be created, and some service workers need to be added, in addition to ensuring some security and speed protocols be followed.

Usage of scalable frameworks For conversion to a full-fledged PWA, more robust and scalable frameworks need to be used, as the current file tree is not at all opinionated, which was good for initial development but can create scalability issues later on.

Getting on-field feedback The application hasn't been tested out yet. Even though serious considerations for maintaining user accessibility was taken, we can't be certain about its working if we don't have feedback from the demographic it's built for.

8 Conclusion

Medical image diagnosis is finding increasing importance in tracking diseases early on, and preventing instances of wrongful detection due to human error. The tools that can help us build automated software already exist and are quickly improving, as we showed in this report. A large community of developers, could

thus, address such issues and help build software that could mean the difference of life and death for the millions of rural and suburban citizens across the country.

We attempted doing the same for our application. The source-code for the same is available on GitHub for anyone to work and improve upon. (code repository is available at https://github.com/SamyakSahu/spasht-app)

Other than that, this project was a great learning experience in terms of getting us exposed to the different web development technologies and popular practices, in addition to the journey of creating a meaningful solution suited to the unique needs of the rural and suburban population of our country.

I would also like to extend my gratitude to Dr Hariom Aggrawal, Dr Vinti Agarwal and Dr Rajiv Gupta for mentoring me throughout the journey and giving their valuable inputs on the way. The project wouldn't have gone nowhere without the sincere efforts of my team members, who worked the image classification algorithms to the shape that was used in this application.

9 Tools used

The following tools were used for the successful implementation of all the components of the application, and also the components outside of it:

Heroku A cloud platform offering Platform-as-a-Service (PaaS). A stateless Heroku dyno was used for hosting the application. The Heroku Command Line Interface (CLI) will be used for all the DevOps for the application.

Figma A wireframing and prototyping application which was used to create and finalize the application design.

GitHub A version-control and source-code management solution which makes use of *Git* (a distributed version control system). The tool was handy in maintaining multiple versions of the code, and provided backup at crucial times of local data loss.

Creately A flowchart creating tool, which was used to visualize the application structure. As opposed to many other tools used for this, Creately was the best suited to do the task at hand, quickly and intuitively.

Notion Is a database-derived note-taking application, which was used to maintain work progress throughout the project. A Notion page would also be used to set up a documentation page for this project, if need be.

Google Docs The online word processor from the Google Suite of business applications was used to maintain work logs in the project.

Overleaf A LaTeX document editing software which was used to create this project report.

Visual Studio Code The open-source text editor from Microsoft, along with special dependencies for Python, Vue, Angular and all the other tools and technologies that were used during the course of development of this application.

References

1. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile Software Development Methods: Review and Analysis, arXiv (2017), https://doi.org/1709.08439
2. Tandel, S., Jamadar, A.: Impact of Progressive Web Apps on Web App Development, IJIRSET, Vol. 7, Issue 9 (2018) https://doi.org/10.15680
3. Web.dev - What are Progressive Web Apps? https://web.dev/what-are-pwas/
4. Web Application Framework - DocForge (archived) https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web_application_framework
5. Angular (Web Framework) - Wikipedia, https://en.wikipedia.org/wiki/ Angular_(web_framework)
6. Vue.js - Wikipedia, https://en.wikipedia.org/wiki/Vue.js
7. FAQ: General — Django documentation — Django https://docs.djangoproject.com/en/dev/faq/general/
8. The Advantages and Disadvantages of Using Django https://datafloq.com/read/advantages-and-disadvantages-of-using-django/3050
9. Foreword - Flask Documentation (0.10) https://web.archive.org/web/20171117015927/http://flask.pocoo.org/docs/0.10/foreword
10. Redmon, J., Farhadi, A.: YOLOv3: An incremental improvement, arXiv (2018) https://doi.org/1804.02767
11. The DCI Architecture - A New Vision of Object-Oriented Programming (2009) https://web.archive.org/web/20090323032904/https://www.artima.com/articles/dci_vision.html
12. Benoni, D., Lavallee, L.:Growth Design: The Psychology of Design-https://growth.design/psychology/