Team Innovaluation

Lucas Myers

Adam Winebarger

Kyle Smigelski

Technical Advisor: Grant Alphenaar

Table of Contents

Overview

Innovaluation TST is a cross platform mobile application that will determine the result of a Tuberculosis TST test based on uploaded images of the test injection site immediately after and between 48 and 72 hours after the initial injection. The application will utilize a machine learning algorithm running on a remote cloud-based server to run image calculations and determine test results.

Languages

For the mobile application portion of this project, the team will be making use of Dart programming language using the Flutter framework for cross platform functionality. This will ensure that we can create and manage Android and iOS-compatible applications through a single codebase.

For the database and web-service portion of this project, Firebase's Cloud Firestore makes use of NoSQL, storing all information as Documents, References, and Collections. In addition, Firebase functions will allow the team to develop server-side logic using either Javascript, Typescript, or Python. While a good portion of the server-side programming may be done in Javascript, the team may be using a limited amount of Python code as well for the integration of Machine Learning tools at the server side, as many of the machine learning tools that are being examined in this preliminary stage are written in python.

Libraries, SDKs, and APIs

Flutter SDK

To allow for speedier development and the ability to deploy iOS and Android applications while managing a single codebase, the team has selected Flutter for this project.

Flutter Package Manager

While the Flutter and Dart standard libraries are very robust, Flutter also includes a package manager that will allow the team to install any additional libraries or dependencies that they may need.

Flutterfire

The Flutterfire package suite offers a myriad of packages for compatibility and ease-of-interoperability between Flutter applications and Google Firebase's Cloud Firestore database service.

Firebase Auth

Firebase Auth is a Google package designed to handle authentication of end-users in Flutter Applications. It works particularly well with the Google Auth module available through Firebase Databases and also handles encryption and storage of user credentials.

Firebase Cloud Functions

Firebase Cloud functions will allow for much more robust functionality of Firebase databases. This will prove particularly useful in the implementation of push-notifications to the end-user application.

Firebase ML Model Downloader **(potentially)**

Firebase ML Model Downloader is a FlutterFire package designed to allow ML models built with Google's Tensorflow_Lite and stored within a Firebase database to interoperate with Flutter applications. Though the documentation suggests that this is a package meant to help ML tools run primarily on the edge, this is a package that still may prove useful in the later stages of this project.

Firebase Cloud Firestore

Cloud Firestore is a NoSQL database service run by Google's Firebase. This database/cloud service was picked over others as multiple members of the team have experience with Firebase services and the open-ended nature of NoSQL databases should prove beneficial in the early stages of development.

While data in Cloud Firestore is normally stored as Documents or References - in sort of a File Manager structure - Cloud Firestore also allows for the storage of images, complex objects, and even has functionality to allow the team to run machine learning tools on the server side, as well as create functions to allow for push notifications to be sent out to certain end-users under certain conditions. Because of this versatility and diversity of tools available, the team feels that Cloud Firestore will prove useful to the development of this project.

Google Authentication

Google Authentication is a module offered by Firebase that streamlines the process of creating users, securely storing their information, and even login/logout procedures done at the end-user level. Because this application will be storing/accessing sensitive medical information of end-users, this is perhaps one of the more essential modules that will be included in this project.

Firebase Functions

Firebase functions will be an essential model for running any backend logic that our project may need. This module will be necessary for allowing Firebase to automatically send push notifications to end-users based on certain conditions; as well as handling any server-side machine learning tools that will be incorporated into this project.

Code Repository Organization

Branching

Our git workflow is composed of 3 branches - master, develop, and features. The features branch will be split into separate branches that each correspond to a Jira user story.

Master

This branch will contain the most stable version of the code for each sprint. All pull requests to this branch need to be reviewed and approved by one or more team members.

Develop

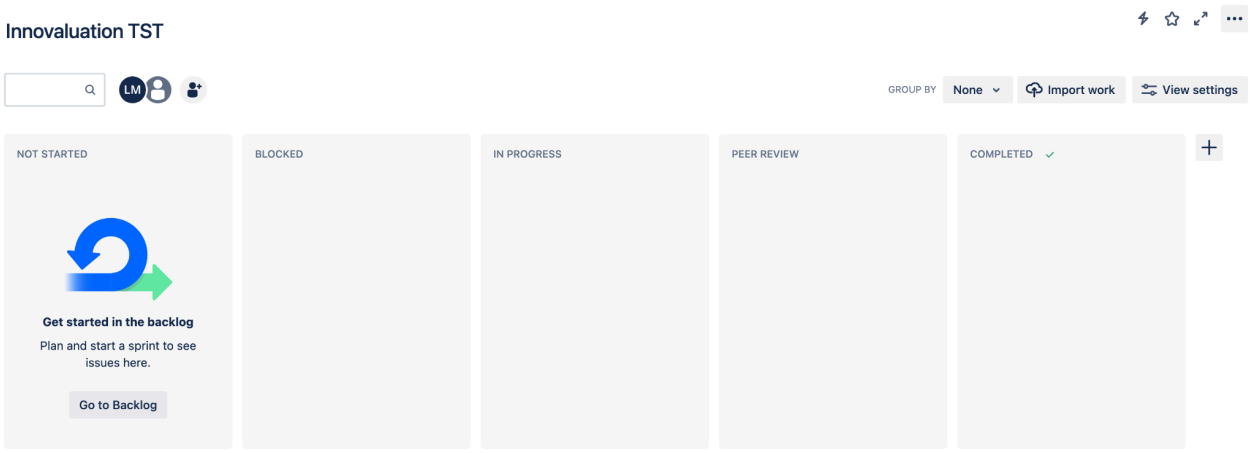
This branch will contain the latest stable version of the code that is currently being developed. All pull requests to this branch must be approved by one or more team members.

Features

These branches will each contain a feature of the code that is currently being worked on. The name of each branch must match the corresponding user story in Jira. All Jira user stories and feature branches will begin with the INO prefix. Once a feature is complete, the responsible team member will submit a pull request to merge their feature branch into develop.

Jira

Our group will be using a standard board in Jira that follows the SCRUM methodology. We will create an epic for each feature we want to implement. Each epic will contain several user stories that divide the feature into smaller tasks.



Backlog

The backlog will contain all of the user stories in our current sprint that have not been started on.

Blocked

Blocked will contain all of the user stories in our current sprint that we cannot continue working on due to a blocker outside of our team's control.

In Progress

In Progress will contain all of the user stories that we are currently working on.

Peer Review

Peer Review will contain all of the user stories that have been finished and are being reviewed.

Completed

Completed will contain all of the user stories in the current sprint that have been completed, reviewed, and approved.

System Organization

InnoValuation TST users will be greeted by a splash screen and a login/signup button using Google Auth. Once the user is authenticated, they will move to the home screen which will display all the information relevant to their TST test, depending on the status.

A new user will be required to complete a questionnaire related to their medical history, which will be stored in Firebase Cloud Firestore. After this is completed, and they have taken their TST test, the app home screen will display the time and date they need to take a photo of their skin test reaction site (between 48-72 hours after their initial injection). At the 48 hour mark, the app will send the user a push notification and display a “Take Photo” button on the home screen, visible for the next 24 hours. Once the user takes the photo, it will be stored in Firebase and sent to the Firebase ML Model, which will process the photo. If it is able to determine a positive or negative result, it will return that information to the user in a push notification and on the homescreen. Additionally, photos taken by users will be stored on Firebase and available for us to use in improving our dataset.

Development

Our application will be developed in our own local environments. To make all of our environments as uniform as possible, we are all going to use the same configuration files, which can be found in our GitHub repository. Because of the cloud-based nature of the cloud firestore database, anyone from the group will be able to send data and/or make changes to the database and view those changes in real time. While this can prove to be somewhat of a double-edged sword in the later stages of development - data will have to be somewhat standardized if the app is to have any hope of making sense of it after all - the freeform approach can still prove quite useful in the early stages of development, particularly in coming up with a structure for out NoSQL database. For the time being, there are no requirements for additional computer hardware as everything will be accessible through either the git repo or the Firebase console.

Non-Trivial Requirements

Our project has three major components: the application, the database, and the ML model. Currently, we are focused on getting the app and its database connections working first, due to the complexity and uncertainty of our ability to implement an accurate model.

Application:

The mobile application portion of the app will likely be the cornerstone of the project as a whole, and the main component in the team's proof-of-concept for this project. This application will be developed using the Flutter SDK in order to achieve iOS and Android compatibility with a single codebase. In addition, the Flutter package manager will be very useful in downloading additional flutter libraries and dependencies for project elements like interoperability with the cloud firestore database and potential machine learning integration. This package manager will also prove useful for any additional dependencies that the team may find are needed in the future. With the exception of the packages mentioned earlier in the report, this app will likely not need much outside of what is organically part of Flutter's standard library.

Cloud-Firestore Database:

Firebase's Cloud-Firestore database function is a cloud-based NoSQL database that, in addition to storing primitive data, allows for the handling and storage of images, user data, and even functions and objects. In addition to storing the users' questionnaire data and images of TST injection sites, firebase/cloud-firestore also includes functionality for things such as the creation and handling of user accounts, generation of authentication tokens based on user UIDs, functionality for push-notifications to the end-user application, and even the ability to incorporate various ML libraries/algorithms into the server-side code. Because of this versatility, the incorporation of firebase into this project will allow for a myriad of capabilities - such as data handling, storage, and manipulation, as well as web-services - that would likely require 3 or 4 tools to achieve the same functionality of tech stacks from similar projects that did not make use of Firebase.

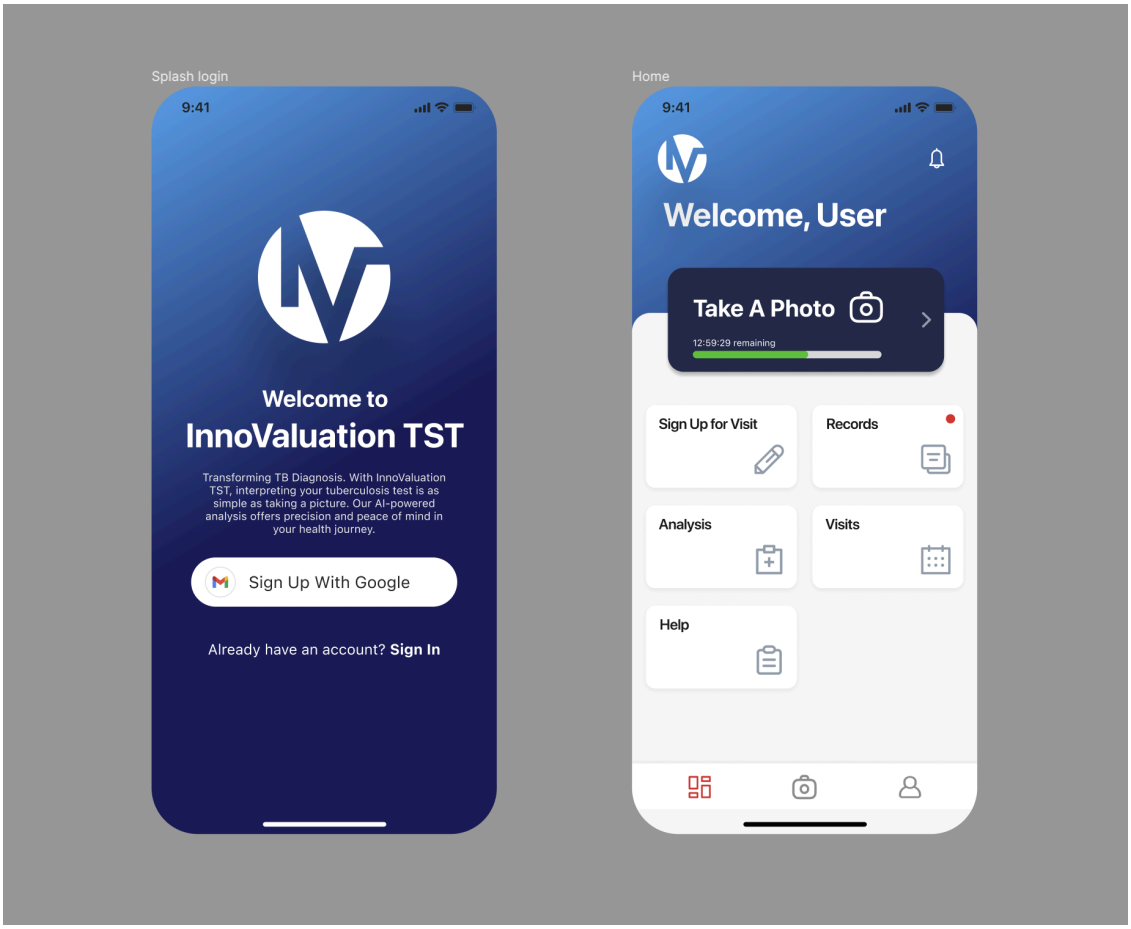
Machine Learning Model:

While the team is not currently certain on the specifics of what machine learning tools, algos, etc. will be implemented, the team is in agreement that, more than likely, this project will require some sort of ML tools in order to achieve the overarching goal of this project. Because of this, the first two sprints will likely involve simultaneously developing the components of this project that will not immediately require ML incorporation, and researching into the most viable/applicable machine learning tools into this project.

Documentation & Styling:

For documentation, the dart compiler includes *dart doc* which will allow for auto-generation of documentation marked with /// or /**...*/. With this tool, the team will basically be able to document as they go. Because this project will likely roll into another semester, extra care will be made to ensure that our code and logic is properly documented. In terms of styling, Flutter has a somewhat unique styling that somewhat resembles HTML code. Because most of what is done in flutter is instantiating objects with a large number of input parameters - and said parameters are often also objects with a large number of input parameters - the HTML-esque styling that is often used by Flutter developers often makes for the most readable syntax.

Mock Ups



Users will be greeted by a login screen that will allow them to sign up with their google account or login. After login, they are brought to the home screen. The “Take A Photo” button is centered as it is the main feature of the app. Below are various buttons:

- “Sign Up For a Visit” will present the user with a variety of links for TST testing

scheduling.
- “Records” will let the user create/enter their medical records in the form of a questionnaire.
- “Analysis” will display results from past photo submissions.
- “Visits” will display the user's past in person appointments.
- “Help” will display a guide to using the app and a link to the CDC website for more information on tuberculosis.

Work Policies

After considering each of our team members' strongest areas of expertise, we assigned general roles that will allow us to work efficiently. Adam will be lead programmer, as he has experience with the libraries and SDKs we are using, and his expertise will be useful to Lucas and Kyle for learning Flutter/Dart. Kyle will focus on mainly the front end and UI of the application, as he has experience in UX/UI design and frontend development. The team will also be receiving help from Grant Alphenaar on the machine learning model and its integration into the application.

Lucas will be managing most of the communication between our sponsor and any other experts we contact for assistance in this project. He will also focus on team management with git and Jira, due to his experience using both at work. All members will contribute to documentation and presentations, as well as research.

Testing

Since we have a very short development period, we will not have much time or focus for testing. However, we will ensure our code is reliable and meets our standards by making sure all pull requests are viewed and approved by at least one team member. The testing we will do will be manual verification of the machine learning model to ensure that it accurately predicts the TST test result based on the uploaded images. We will also be checking in with Grant Alphenaar and Roland Heusser, two GVSU alumni and professional software developers who are partnering with the University for this capstone course, to ensure that our code is safe, sensible, and reliable.

//Since we are going to be using SCRUM as our workflow methodology, there isn’t going to be much time or focus spent on testing. Instead, we are going to make sure our code is reliable by making sure all pull requests are viewed and approved by at least one team member. In addition, the team will periodically check in with Roland Heusser, a software developer partnered with the University for this capstone, to ensure that safe and sensible coding practices are being adhered to.

Special Considerations

Similar Studies

During the early phases of conducting preliminary research into the viability of this project, the team found a few studies conducted from other universities into positive tuberculosis identification by means of a mobile application and Machine Learning algorithm. One particular study of note, one by the Birla Institute of Technology and Science in Pilani, India (or *BITS Pilani* for short), looks to have achieved something similar to this project. Even so, the team still feels that this is a viable project and finds that reaching out to these other universities may

Synthetic Image Generation

At the recommendation of Grant Alphenaar - our resident Machine Learning expert - the team is considering making use of synthetic image generation in order to generate a large amount of training data with a relatively small number of legitimate TST results. This is not something that has been well tested in the past, and it certainly would not be ready to go to market at the end of this. But for an early-stage proof-of-concept product in the given time that we have, this should prove to be viable and it should also hopefully shed some insights on the viability of using synthetic image generation to train ML models on a larger scale

Inducing False-Positive TST Result

One other problematic area in securing training data for a TST detector is obtaining images of positive TST results. Fortunately, tuberculosis is not a super common occurrence in the United States. This, however, presents a somewhat unique challenge for Team InnoValuation as securing images of positive TST results to train the model on will prove somewhat difficult. However, there is one somewhat unconventional way in which the team could secure positive results.

There exists a somewhat common set of conditions that induces a false positive result within the TST test. Because test results are considered invalid if more than 72 hours have passed since the initial injection, and because a follow-up TST injection will have to be performed in order to get valid results, it is fairly well-documented that follow-up TST injections that are performed to close to the original lead to abnormally high false-positive rates. Because of this, combined with the fact that tuberculosis is not super common in the United States, the team feels that deliberately inducing a false positive result through this method will be the best way to secure positive results for the training data/synthetic image generation. However, this is under the pretense that there are no lasting side-effects with deliberately inducing the false-positive. Further consultation with Dr. Steven Triesenberg and Linda Chamberlain will be needed before the team can move forward with this strategy.

Physical Medium to Track Test-Site Delta

In order to help the ML model to better track growth that occurs at the TST injection site, the group feels that some sort of physical marking on the subject will better help the ML model to track scale and distance on individual photographs. While the team is still in the process of refining the specifics, the current idea among the team is to place a “temporary tattoo” in the shape of a square or bounding box around the edges of where the injection site will be, then taking photo a photo after the injection and a subsequent photograph 48-72 hours after the TST injection to track growth.

Lack of Available Training Data

One of the major obstacles of this project is the lack of any sort of training data generated prior to this project. While the team has considered a few ideas on how to obtain sufficient training data on which to train an ML model, most of these would require a review from an Institutional Review Board (IRB). Given the time it would take to get IRB approval on any of these acquisition methods, most of these will not be realistically achievable within the allotted time that the team has for this project.

As such, the team is exploring some potential solutions that address this problem. The first will be to fork the end-user application into two: one that does collection of training data through the end-user’s camera (with consent, of course), and another with a placeholder ML model to demonstrate the proof-of-concept for the project. Another involves making use of saline solution in order to induce growth that would look similar to a TST in order to give the team some fake positive results to train our data model on, while another still involves trying to repurpose a general purpose machine learning model to fit our needs. Currently, team InnoValuation is still looking into the viability of these solutions and others. But the final decision on implementation of machine learning components will be determined after a baseline end-user application is created.

Deliverance and Milestones

Sprint 1

- Create baseline application that roughly resembles what the end-user will interact with
- Install necessary packages/dependencies to mobile application
- Set up Google authentication on the login portion of the mobile app
- Initial Setup of Cloud Firestore Database
- Get app and database talking

Sprint 2

- Ensure that end-user app is in a good “baseline” state.
    - Make sure repo is marked in such a way that it is apparent that is the commit to roll back to if something goes wrong in the future
- Component testing of end-user application (if there’s time)

- Begin integration of ML algorithm into database
- Also serves as a buffer sprint in the even that anything from sprint 1 is left unfinished
- Potentially begin training of machine learning algo

Sprint 3

- Complete integration of known ML tools into firebase server
- More robust training of machine learning algorithm here

Sprint 4 - contingency sprint

- Buffer sprint
- Polish GUI
- Bugs

- If everything from the first 3 sprints completed
    - Focus on documentation, finding bugs, and improving UX/UI for end,user app
- Else
    - Make use of this sprint to get caught up, then proceed with the above
- In any case, will also likely need to continue training the model in this sprint
    - First cases of live testing should also be done here if they were not started in an earlier sprint

Sprint 5

- Documentation / Polishing

Sprint 6

- Presentation and product submission