Phase 3
The At-Home Covid Test with Smart Capabilities

Spring 2022

Group-8

Prepared by: Saul-Leal Garcia Justin Phuong Mark Martin
Damandeep Singh Dean Quach Can Chai Jared Mallari

Table of Contents

1. Project Overview

We are pleased to present our covid testing device with smart capabilities. The primary objective of the device is to allow a customer to take a covid test with the capabilities to send test results over to a user interface available in the form of an app or website. The user interface will have the capability to send their test results to the government or non-government institutions. This comprehensive document includes requirements, design, testing, security, resource, budget, and a maintenance plan for the at-home covid testing device.

2. Requirements

We have divided our requirements into hardware and software requirements. Hardware requirements are listed as a table and software requirements for the user interface are given through a user story.

Req ID	Requirement Description
1	The device should have built-in six silver oxide batteries (1.55 volts) as a power source to the device
2	The device will have on and off buttons to enable and disable the device
3	The device should have a bluetooth range of upto 800 ft
4	The device should turn off after 5 minutes of inactivity
5	The red LED light should turn on with positive test result
6	The blue LED light should flash unlimited times until a device is paired.
7	The green LED light should turn on with negative test result
8	The device should connect to only one end-user personal device at a time through Bluetooth
9	The device should take no more than 10 minutes to process covid results
10	The device should send a matching 5 digit pin to ensure the connection is established between the user's choice of personal device and the covid device

Table 1. Detailed Requirements of Covid Device

2.1 User Story (App)

Bob is a new user and have recently completed a covid test from the hardware component.
Bob can go into google play store or apple store and download our application. Bob should also choose whether this app can access mobile device data or not when he enters the app for the first time. After installation is complete, our app will display options such as if they are a new user or an existing user. If Bob is a first-time user, he should press on the “New User Registration” button, otherwise the “Existing User” button. Bob will be in our U.I. when registration is complete, which will display options such as connecting the device via Bluetooth, sending test results to authorized health providers, government and non-government institutions, editing profile, and history of current/previous covid tests. Once Bob connects to the covid device via Bluetooth, a 5-digit pin will appear on the U.I to confirm that the correct device is requesting to connect with the user. Once confirmed, Bob will have access to the analytics, heatmap, and news tabs. The covid result data will transfer to the app, and under the analytics tab, a personalized covid results report will be generated, which includes: name, address, social security number, phone number, home address, test result, and date/time. Under the same analytics tab, Bob may send the data to their requested institutions using either institution’s name or use the unique institution code and plan to send their data within 24 hours of the report generation.

2.2 Added Requirements

In this phase we added new requirements to our developing phase to ensure that we can produce a reliable product for our users. Our reliability requirements for our entire system will cover most of the important cases such as ensuring the app and our main components will have a long life cycle for over 1 year so that we can continuously support the use of this product. We will increase the performance of accessing the app via using face id or passcode. We will increase the speed of data transfer from the device to the database. We will also improve our security for the website/app and encrypt the data that’s sent to it just in case there are any attacks.

2.3 Methodology

To fulfill development of our product, we will apply a mix of the agile and waterfall methods.

- Our physical device is primarily hardware, and as a result, will benefit from one projected development cycle. For this reason, we will be employing the waterfall method in development of the physical COVID test, including the LEDs, detection system, and bluetooth elements.

- Because our database will likely be designed once with minor updates, we will be applying the agile method for development.

- The mobile application with analytics portion of our project will have to be updated periodically, both due to the possibility of novel COVID strains and updates for user feedback. To accommodate these regular changes, we will be using the agile method.

3. Design

3.1 Design Specifications

The design of the at-home covid test device is split into two subsystems. The first subsystem is the covid test hardware component and the second subsystem is the application/website software component for users to interact with their respective data interested parties. Here is an abstract level use case diagram that describes the interactions between the user and different actors:

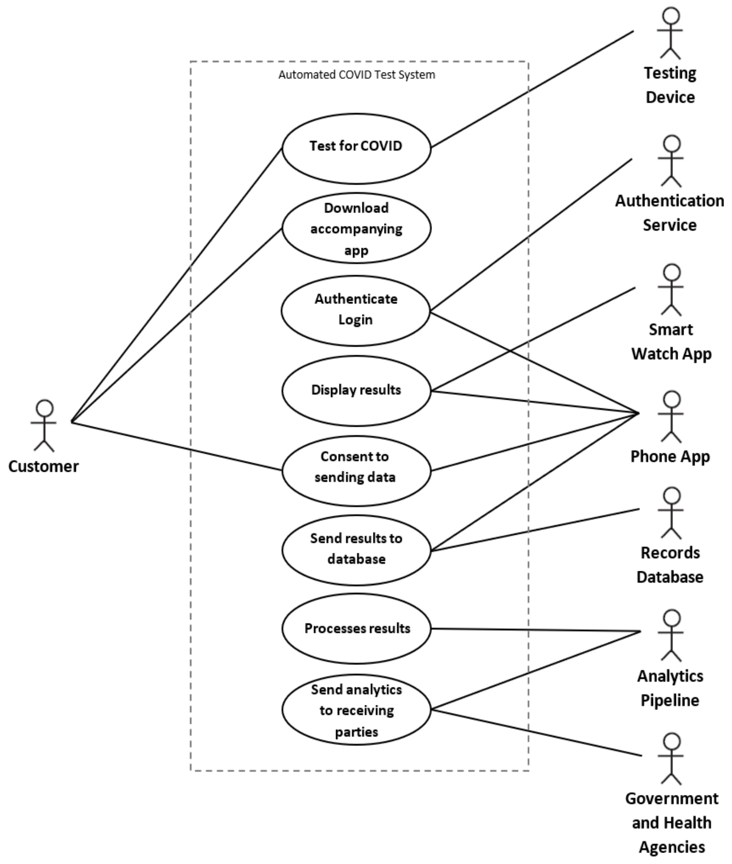


Figure 1. Use Case Diagram of Whole System

The following sequence diagram explains the detailed interaction and procedure of how the whole system will work in a stepwise manner with the integrated system (subsystem I and subsystem II):

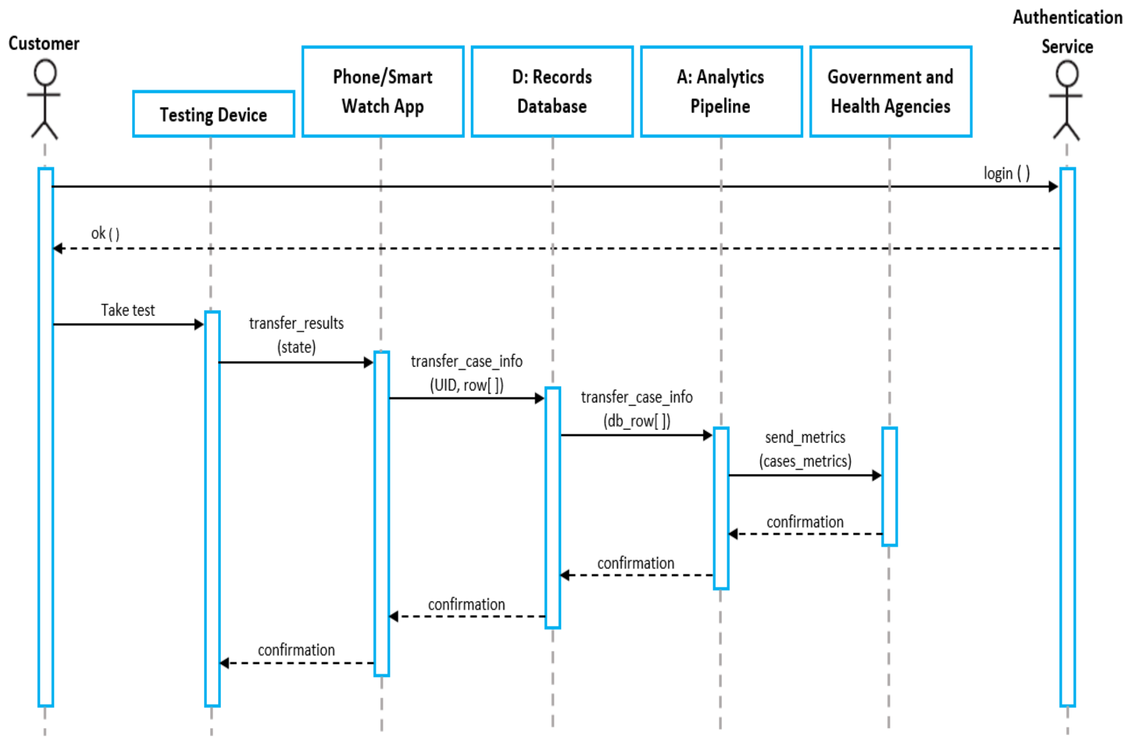


Figure 2. Sequence Diagram of Whole System

3.2 Subsystem I: App/Website

Our project prototype relies on the creation of an easy-to-use user interface for our customers. The main functionality of the application/website is to provide the user with capabilities to personalize their covid results along with the privileges to interact with institutions requiring immediate results of covid tests.

3.2.1 User Interface

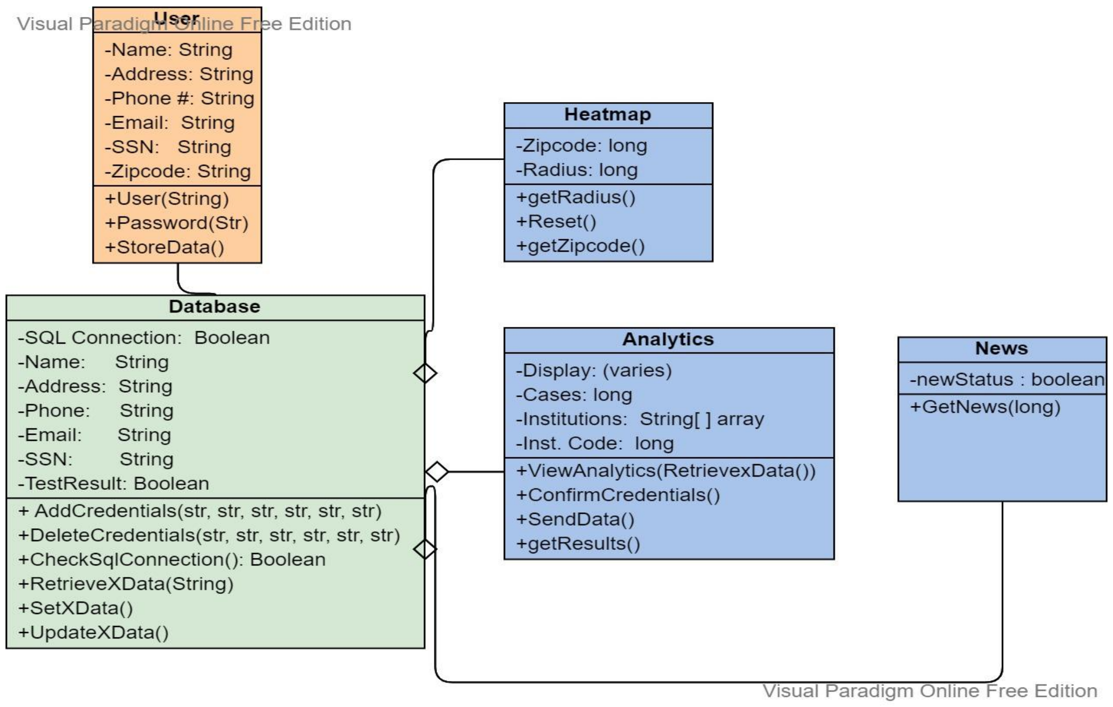


Figure 3. Class diagram of the app/website

The app and the website will consist of the four classes: user, heatmap, analytics, and news. The user interface of the app/website is similar in layout and functionality for all users. A relational database is used to store and retrieve user data to execute requested operations. **Changes:** Introduction of passcode protection system for the app and website to provide robust protection against attacks.

User Class

The purpose of the user class is to provide the registration page for the new users and allow them to add their personal information such as name, home address, phone number, email, social security number and zipCode (optional). The user class will communicate with the relational database SQL and execute CRUD operations to validate the data into the system. A passcode protection system to allow users to sign in into the system.

Private variables: name, address, phone number, SSN, and ZipCode (all string data type)

Public methods:

User(string): Automate a username creator or allow user create their own
Password(string): Automate a password generator or allow user to create one
StoreData(): communicates with database to store the user input into a table

Passcode protection: The user may generate any 4 digit pin to access the user interface. After three failed attempts, the user interface will lock for 10 minutes.

Heatmap Class

The purpose of this class is to display a heat map according to the area's zip code. The heatmap class will use some current or previous datasets present online through the Johns Hopkins covid resource center website. We use the streamlit library of the python programming language and generate a heatmap as per the data. The map will allow users to know the high risk and low risk areas around their neighborhood. The user can also increase the radius of their area using the radius feature. An additional feature of reset is provided in this class to allow the user to reset the heatmap.

Private variables: radius: long and zipcode: long (in case it is not provided)

Public methods:

getRadius(): communicates with the database and gets radius input by the user.
Reset(): resets the current heatmap (a void function)
getZipCode(): communicates with the database and gets zipcode input by the user.

Analytics Class

The purpose of the analytics class is to allow users to access, view and send their sensitive data such as the covid test results. This class will communicate with our database to retrieve the test results that are provided through the bluetooth capability of the at home covid test component of the whole system. The class will also have a list of institutions (government/non-government) that the user may want to send their data to along with a unique institution code to ensure the data is sent to the selected entity.

Private variables: Display, Cases, Institutions, codes (varies, long, string[] array, long)

Public methods:

ViewAnalytics(RetrieveData()): Displays the covid results along with the personal information

ConfirmCredentials(): A void method that will require the user to input their username/password again for security purposes.

getResult(): This function will display the test results to the user (void function)

SendData(): We will create a hashmap of where keys are institution codes and values are institution names for a faster lookup functionality. This function will also carry the ability to send data to the institutions through an email (we assume institution emails exist in our database) while invoking the getResult() and RetrieveData() methods for a report generation.

News Class

The news class aims to display covid results related news according to their zip code and recent CDC and government updates on the mask and covid surges. The news class will notify the user regarding covid surges in the neighborhood.

private variable: zipcode

Public method: getNews(zipcode) → calls the zipcode function from the database and the google news API to display local news updates and outbreaks.

Database

The database plays a critical role in development of both hardware and software components of this system. We are planning to use a relational database to store all the necessary variables and features mentioned in the class diagram of the app/website. The schema for the database will have the same variables listed in Figure 3 and mostly all the variables will be private and abstract to add a security layer. The database will have many functions for the data:

- CRUD operation queries for basic create/read/update/delete the structured table with variables of the multiple class diagram
- SQL boolean connection output to ensure its communicating with the app/website

The combination of the four classes and the database work interchangeably to provide an easy-to-use interface for the customers interested in sending their covid data to the interested parties. The user may have a privilege or choice whether they want to send their data to the interested or wish to keep the data on our database for future references. Another requirement for our database is to authenticate the user details if the user already exists in the database. We will have a SQL query that may search for the social security number in the database and look up the matching name input by the user at the time of registration. If the existing user's data does not match, the user will not proceed to the next step of accessing the dashboard.

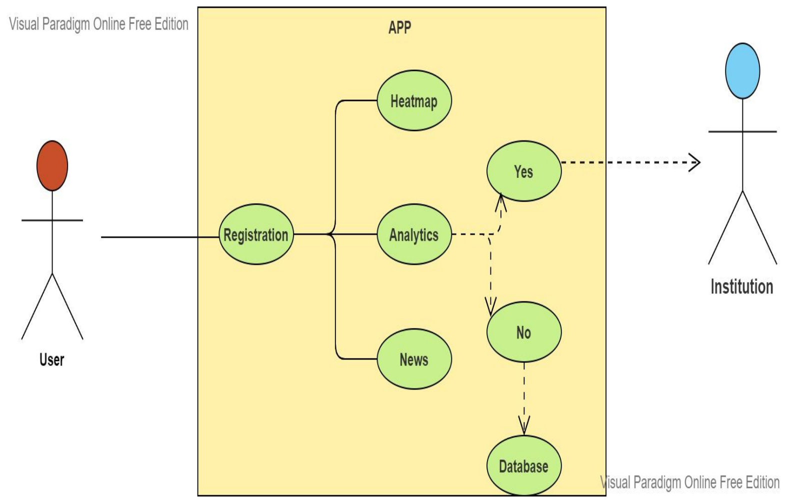

Visual Paradigm Online Free Edition
APP
Heatmap
Yes
Registration
Analytics
No
News
Database
User
Institution
Visual Paradigm Online Free Edition

Figure 4. Use Case Diagram of App/website

The user will have simple steps over the app/website component. First, the user must register with their personal information to allow the successful registration and transfer of data over Bluetooth. Second, the user can select one of the three options: heatmap, analytics, or news. Third, the user can choose to send their covid test results to the institutions or avoid sending their covid tests results; however, the results are stored in our database for quality purposes.

3.2.2 Technology Stack

One of the primary goals for our project is to develop a web application and an app to promote personalized interaction between users and the data-interested institutions. Both website and app will have similar functionality and layout as both software components will have the same features/variables and methods to attain a standard level.

The whole website will operate on the HTML5 and CSS3 technologies. In addition, many front-end and frameworks are used to develop a vivid and easy-to-use user interface for our customers.

Here is a list of technologies we are going to use to develop our website:

- Frontend technologies: HTML5, CSS3, Animation.CSS, and Bootstrap
- Backend technologies: Javascript/EJS, React.js, Node.js, and AWS Relational Database Environment (SQL)
- Heatmap development: Streamlit library available on python 3.x to generate a heatmap
- News: Google API

Here is the technology we are going to use to develop an IOS app:

- IOS app development platform: xCode using the Swift programming language
- Android app development platform: Kotlin in Android Studio

3.3 Subsystem II: Covid Device

The hardware device will have the Bluetooth 5.0 transmitter (Tx) to send covid results over to the device. The device will have on/off buttons to turn on and off the device. The device will have three 5mm LED lights: red light (positive result), green light (negative result) and blue light (Bluetooth status). The device will have a place to insert tested cotton swabs for testing purposes. The detailed requirements are also mentioned in the 2.1 section. Since, waterfall methodology is used to design this covid hardware design we will have the chances to accommodate changes earlier into our design as new covid variants emerge.

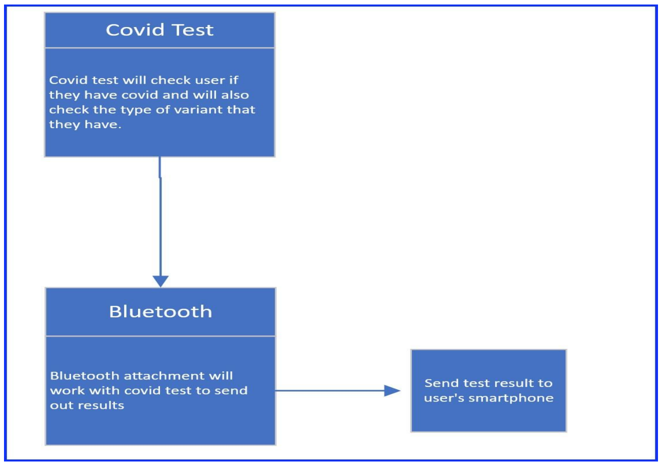

Covid Test
Covid test will check user if they have covid and will also check the type of variant that they have.
Bluetooth
Bluetooth attachment will work with covid test to send out results
Send test result to user's smartphone

Figure 5. Class diagram of the covid device

The covid test device is a physical component, and the communication between the covid device and the software is critical to ensure results are communicable to the right person. We have taken measures to ensure that only one device simultaneously communicates to the covid device. We have enabled a 5-digit pin confirmation method on both the user's device and covid device to ensure the suitable device is connected and no other Bluetooth device is connected to the device and receiving any results. The device will have an embedded timer not exposed to the user to allow the device to reset the device after 5 minutes of inactivity. The device may have a BlueTooth connection range in the covid device to the personal device of 800ft.

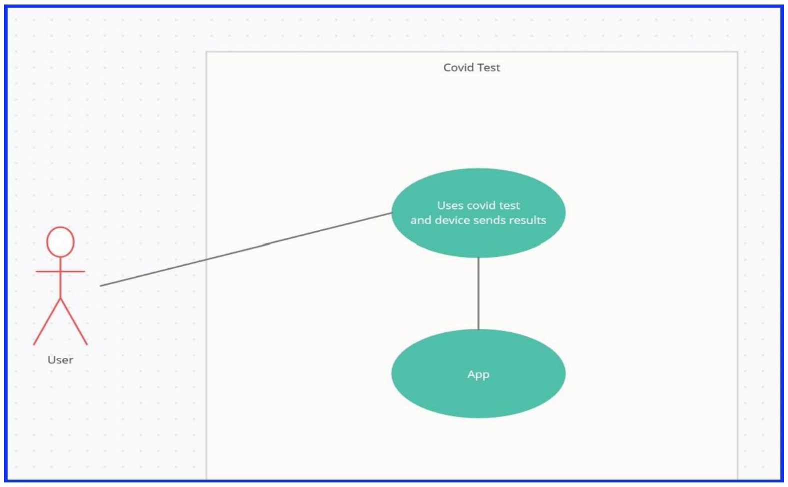


Figure 6: Use Case Diagram of the Covid Device

For the covid device subcomponent, the interaction between the user of the covid test device and the app or website will allow the user to render their results to interested parties. There is only one actor to interact with the hardware covid device to get their results.

4. Security

One of the major priorities of our system is security since our product will contain customers' sensitive information. To save time and money, we will be reusing certain security assets, but we will also ensure that this does not negatively impact either security or the reliability of our system. To ensure security and reliability, we will conduct numerous tests and hire security consultants to ensure our system is up to standard. Our primary strategy to ensure that our system is adequately safe is testing our security and improving upon it. Our budget and timeline demonstrated that we had set aside a large portion of our time and budget to test our security system. This will ensure that our security meets the basic requirements and handles some of the more common cybersecurity threats.

4.1 Security: Reliability

One of the main priorities for our security is making sure that the system between the developer and end-user is reliable. Also, we want to make sure that our product is consistent and failure-free 99.9% every time it is used. To make our system reliable, we implement data encryption to ensure that any information regarding the account info, database, app, and the covid device does not get attacked. We can also use extra layers of protection like firewalls and safeguards to increase security measures. Furthermore, our entire system should have a life cycle for over a year to maintain continuous support, increase web app or mobile app performance, accessibility online or offline, and data transfer.

4.2 Security: Resiliency

We should assume that human errors will occur during system operation, therefore we have to think about defenses and barriers to human error that will be in the system. We will include ways to validate all inputs as a technical defense and to make sure our system is fully functional. We will increase the amount of defensive layers, further making it less likely that we have any holes that will cause a system failure. In the case that the entire system does fail, we can direct the users to the nearest COVID testing center, so that we still give available resources for them to use.

5. Test Plan

The testing of our whole system will revolve around the four testing methods to ensure our system is error-free when our customer is ready to take their covid test. Overall, we may test for various functionalities with some of the high-level testing requirements and procedures, which includes:

- Exploratory testing will be will be executed once build is ready
- Test team will be provided access via VPN
- Test case design will be performed by a QA team
- Defects will be tracked through Atlassian, Jira and given to test team
- Performance testing is considered in later cycles of testing (Q2, Q3)
- Testing will be focused on meeting the business requirements, quality, and cost efficiency
- Testing will be a repeatable, quantifiable, and measurable function of activity
- There will be an entrance and exit criteria
- Testing will be defined but have room for flexibility depending on changing requirements

5.1 Unit testing

We will perform unit testing on both the software and hardware components of the system. This is our first stage of testing. For our software component, we will use the following functions to test the integrity of our app and the website:

- func setUpWithError(): checks for valid characters or inputs from username, password, address, social security, email, zip. Exits upon invalidation returns false
- func validateResults(): checks for valid results based on known COVID test criteria
- func checkUploadTarget(): checks for valid targets to upload results. Based off of known health networks and facilities
- func checkDownloadData(): checks for valid downloaded data from database and ensures complete information with no null values
- func isRepeatedPassword(): checks if the password is repeated correctly upon creation

For our hardware component, which is the covid device, we will test for the multiple units or sub-components of the system using the following functions:

- Manually check ON/OFF buttons to ensure the device shows the expected behavior.
- Check Bluetooth range of the covid device:
    - Checking the Received Signal Strength Indicator (RSSI) value of -40 to 55 and no less than -70 to ensure a strong connection between the covid device and the personal device.
    - The RSSI value is available under the bluetooth option of many smart devices.
- Check voltage of batteries should have ~1.55 volts/battery using the electronic voltmeter.

5.2 Interface Testing

The interface testing will mainly focus on the bluetooth and database connectivity of hardware and software side.

- Testing between the web application and mobile application interfaces will be conducted

- Testing between the database and web application interfaces will be conducted

- Testing between the database and the mobile application will be conducted

- Testing will be done to ensure strong network connectivity between database and application environments

- Interface between devices and COVID test kit will be thoroughly tested for connectivity, response time, and accuracy

- Bluetooth connectivity between the device and the app results in a higher RSSI value of -40 to 55.

5.3 System Testing

Testing the combination of subsystem I and subsystem II requires integrating the two systems and executing a full scale test on the whole system and expect desired results.

- Testing of usability of the interface for users will be done

- Testing under load stress with thousands up to hundreds of thousands of users will be tested for capacity limit testing

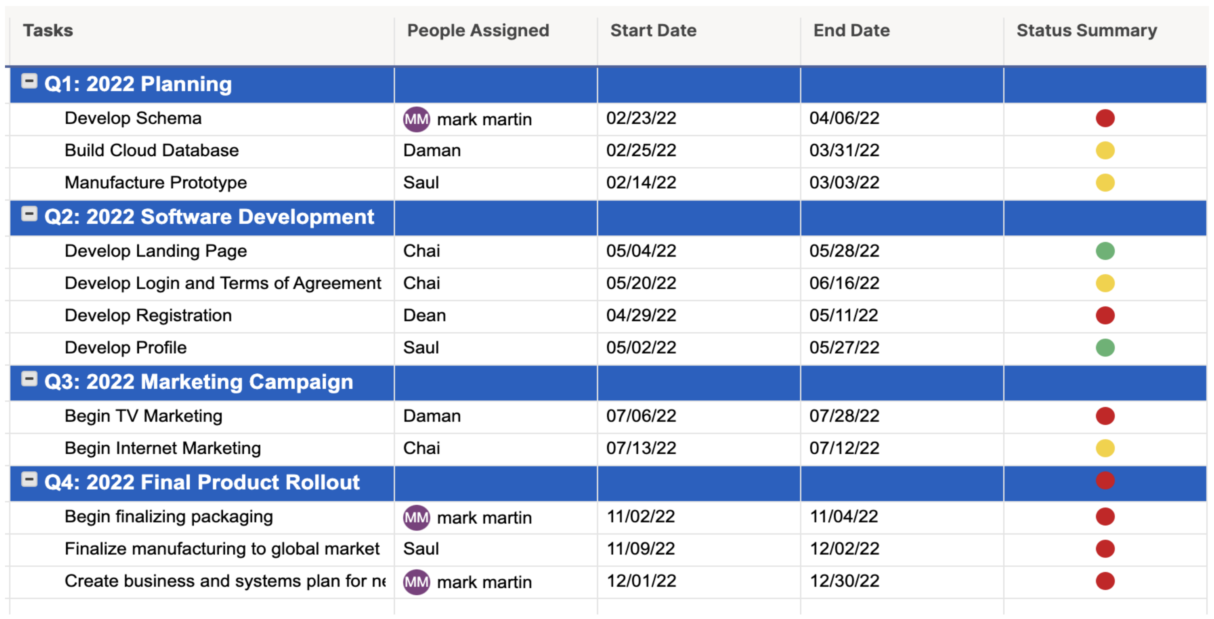
- Testing for network connectivity between interfaces will be conducted
- Crash testing will be conducted to ensure software is fully functional after any hardware disfunction or software crashes
- We will be using the Eggplant software to seamlessly integrate CI/CD pipeline and compress testing cycles to allow for quicker releases

5.4 Release Testing

Our last stage of testing to identify any bugs and errors in the software and hardware component and make sure the product is ready to reach our end users.

- A suite of release testing programs will be running before release to ensure that high priority bugs are non existent in the program
- Specific release tests include working notifications to the user incase their data upload went wrong, invalid information on user login or registration, and inability to download data from database
- Testing will also be done before release for proper encryption and decryption of user information
- Stress testing will be incorporated to monitor performance and ensure we can account for 1000 downloads/transactions per second

6. Resources



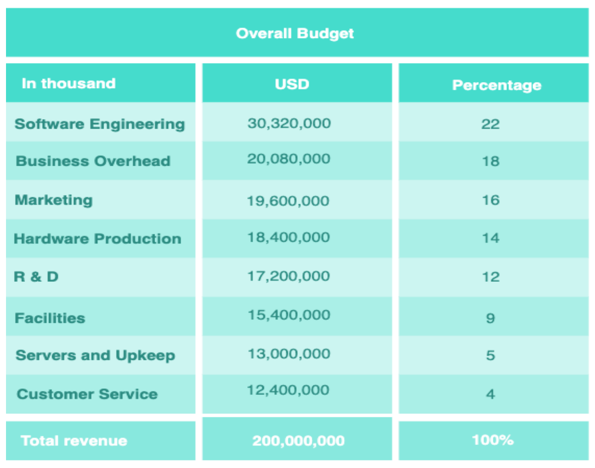
Tasks	People Assigned	Start Date	End Date	Status Summary
Q1: 2022 Planning				
Develop Schema	MM mark martin	02/23/22	04/06/22	🔴
Build Cloud Database	Daman	02/25/22	03/31/22	🟡
Manufacture Prototype	Saul	02/14/22	03/03/22	🟡
Q2: 2022 Software Development				
Develop Landing Page	Chai	05/04/22	05/28/22	🟢
Develop Login and Terms of Agreement	Chai	05/20/22	06/16/22	🟡
Develop Registration	Dean	04/29/22	05/11/22	🔴
Develop Profile	Saul	05/02/22	05/27/22	🟢
Q3: 2022 Marketing Campaign				
Begin TV Marketing	Daman	07/06/22	07/28/22	🔴
Begin Internet Marketing	Chai	07/13/22	07/12/22	🟡
Q4: 2022 Final Product Rollout				🔴
Begin finalizing packaging	MM mark martin	11/02/22	11/04/22	🔴
Finalize manufacturing to global market	Saul	11/09/22	12/02/22	🔴
Create business and systems plan for ne	MM mark martin	12/01/22	12/30/22	🔴

Figure 7: Resource

7. Budget

The projected costs for the whole project is $200,000,000.

Projected Budget



Overall Budget		
In thousand	USD	Percentage
Software Engineering	30,320,000	22
Business Overhead	20,080,000	18
Marketing	19,600,000	16
Hardware Production	18,400,000	14
R & D	17,200,000	12
Facilities	15,400,000	9
Servers and Upkeep	13,000,000	5
Customer Service	12,400,000	4
Total revenue	200,000,000	100%

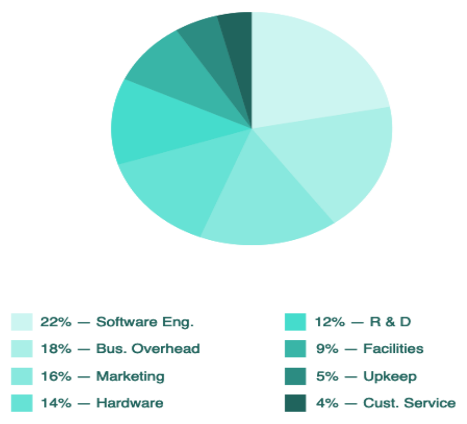


Figure 8: Projected Budget

7.1 Schedule Update

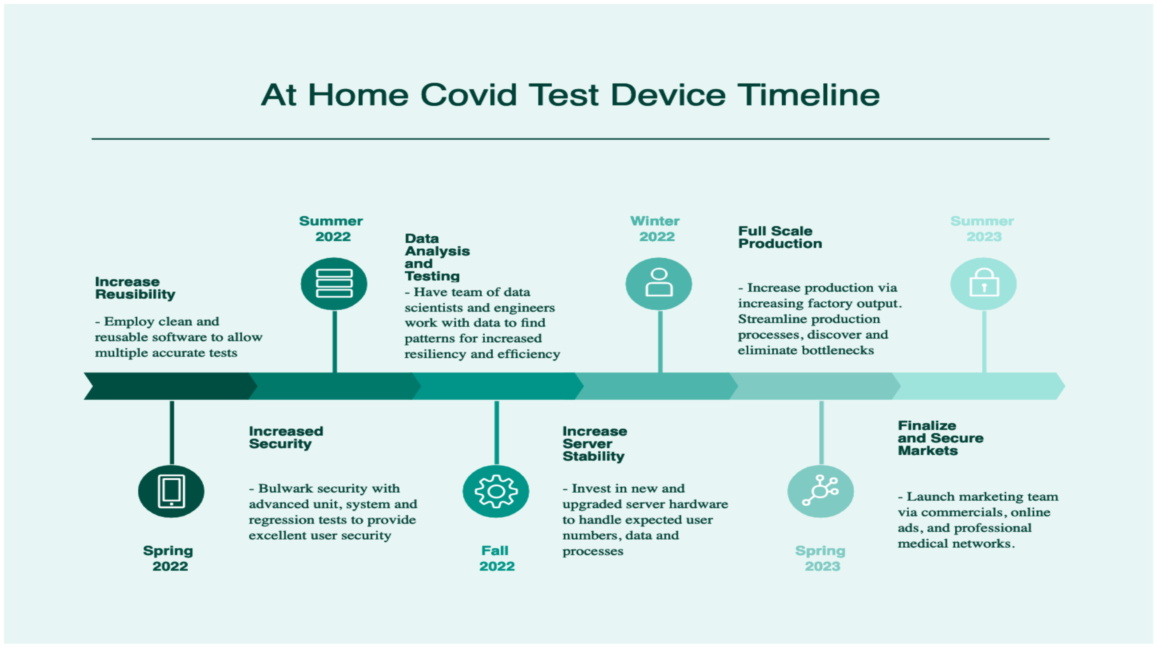

At Home Covid Test Device Timeline
Increase Reusibility
- Employ clean and reusable software to allow multiple accurate tests
Spring 2022
Summer 2022
Increased Security
- Bulwark security with advanced unit, system and regression tests to provide excellent user security
Data Analysis and Testing
- Have team of data scientists and engineers work with data to find patterns for increased resiliency and efficiency
Fall 2022
Winter 2022
Increase Server Stability
- Invest in new and upgraded server hardware to handle expected user numbers, data and processes
Full Scale Production
- Increase production via increasing factory output. Streamline production processes, discover and eliminate bottlenecks
Spring 2023
Summer 2023
Finalize and Secure Markets
- Launch marketing team via commercials, online ads, and professional medical networks.

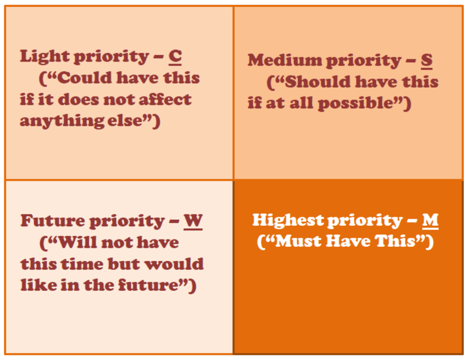
Figure 9: Schedule Timeline

8. Maintenance Plan

The budget for maintenance is $5 million / year, which is about 50% of the software development cost. We plan to release routine changes monthly. To fix bugs/make upgrades regularly will help our product remain competitive in the market without disturbing customers too much. Once a critical problem is identified, our group will work on it immediately and update the new release after it passes the test.

8.1 Changes' Priority

We use the *MoSCoW method* to determine the changes' priority, which have four levels shown below:


Light priority – C ("Could have this if it does not affect anything else")
Medium priority – S ("Should have this if at all possible")
Future priority – W ("Will not have this time but would like in the future")
Highest priority – M ("Must Have This")

1. **M**ust have: The product must have this attribute or feature, it is non-negotiable as the system cannot be used without this feature.
2. **S**hould have: If possible, the product should have this feature. It has high business value, but (if necessary) the product could function without it.
3. **C**ould have: The product could have this feature, but it is less critical. Oftentimes a "C" label simply

indicates a nice-to-have feature.

4. **W**on’t have: These features represent the least *business value* – and will not be included now, but in the future.

8.2 Life Cycle

We want the app and our main components to have a long life cycle for over 1 year. Our product will remain extremely useful to everyone during this pandemic due to its ability to adapt to new variants. Besides, our product software and hardware can be applied to future pandemics and because of its versatility it can be used for different purposes in the future.

8.3 Acceptance Testing

To ensure our released product works at an acceptable level, we created multiple tests to test the efficiency and accuracy of our system. Our system had been developed through two separate development methods, agile and waterfall. As a result, we applied two different methods for acceptance testing.

For our application and analytic pipeline software components, which had been developed with an agile process, we conducted acceptance tests through a series of automated tests. Tests included new tests written at each development iteration and old tests written for previous iterations. This ensures that not only are new changes tested for acceptance, but also ensures that new changes don’t break the previous acceptance of the system. Because we had no direct customer to work with in the development of our acceptance tests, we created tests while keeping in mind what constitutes an acceptable system in the perspective of the consumer. These acceptance tests include automated tests for the following:

- Speed, consistency and security for login methods
- Useability and uptime for the user application
- Security for data sent to the database
- Speed and accuracy for data moving through the analytics pipeline

We found that software systems were efficient and reliable, passing all automated tests created through development.

For our testing and bluetooth hardware components, which had been developed with a waterfall process, we created acceptance tests and gathered results after completion of the product. These acceptance tests included the following:

- Accuracy and speed for sensor detection
- Consistency and speed for bluetooth app connection

We found that our tests were able to detect for COVID with a 99.5% accuracy. This includes both correctly determining positivity and strain. We have also gathered an error rate of 0.01% for our bluetooth component. This includes correctly connecting to the app and correctly transfering results to the app.

Overall, we concluded that both our software and hardware components passed our defined acceptance tests and are ready for the public market.

9. Conclusion

Our project will help make COVID tests much more convenient, allowing users to test and share results with ease. When going through the testing process, we ideally try to put the software through various situations that may “break” the system, further making adjustments to ensure a functional U.I. We plan to release this project to the public while still evolving the system to meet new requirements and improvements. We will continue to adapt our product for any covid variant or viruses that may develop in the future. We will expand the testing capabilities to test for other transmittable diseases. We plan to continue updating and expanding the security of our software to make it as secure as possible.