**Week 2: Progress Report**

**Apurba Das**

My primary concern in Week 2 was to find accessible APIs and data, the usefulness of which would make it possible to detect the presence of phishing links and malicious websites. I began by checking various sites such as PhishTank, OpenPhish, Google Safe Browsing, and VirusTotal. First, I had a tendency to use PhishTank because of its popularity and popularity in academia. When I tried to register, however, the PhishTank site was not allowing new accounts to be created, and this killed our intention of using it. This omission was something that needed to be revisited by us, and we, in the end, re-evaluated our strategy of realizing API integration. Upon discussing the problem in the team, we agreed to proceed with OpenPhish, as it does not use user authentication, and it is simpler to get the data with the help of Node.js.

I also started investigating the process of incorporating Google Safe Browsing into our backend system. It needed a Google API key and was structured with regard to requests. I have been able to write a test script that issues requests to Google Safe Browsing and successfully receives responses from it, and the outcome was encouraging. I captured the structure of the output and the way we could think of it in our UI to present to the users.

I also worked with Aman and Hemant to write up a backend route structure and talk about how we will join together results of several APIs into a unified response, outside of work on the API research. This would make us able to give a user an overarching scanning report as per the multiple threat flunking sources.

Finally, I also configured the root file structure of the backend and put in some API testing routes. I had made sure that the server was working properly and had also configured the scalable development we were in the process of developing. In general we had an initial hiccup with PhishTank, but Week 2 went out well with a definite direction and plan of where the backend scanning is going.

**Hemant Adhikari**

During Week 2, my main task was to start building the frontend structure with Next.js and assist in setting the basic page structure of our application. I also worked with Aman and Apurba to complete the structure of the project, such as the number of primary pages we would require: home, link scanner, email checker, quiz section, log in, and signup. After discussion, we began developing the home page consisting of primitive UI and navigation.

The routing configuration of the Next.js App Router was one of the initial technical assignments that I assumed since we needed an orderly arrangement that would enable us to leverage shared modules such as headers and footers. I also ensured that every page can be accessed through the assigned path, and we also adhered to the regulations of good file structure. To have a clear and steady layout, I also applied some basic styling with Tailwind CSS.

Then I began to incorporate the interface form designs of the link scanner. This included the formulation of an input area whereby users will be able to insert a desired link by means of pasting it into the input area and sending it to be scanned. The form at this point simply wrote changes to input on the console since the backend was not finished, and this setup provided the basis of future API connections.

Besides the frontend, another task I had to perform was to scout alternative email phishing detectors. I did some research on different combinations of words that phishing emails tend to use and found out the possible open-source data on phishing email classification vs. legitimate email classification. Some of the major characteristics that we could utilize are recorded by me, which include suspicious phrases and urgency as well as frequency of keywords.

At our weekly team meeting, I suggested that we implement a training block where the users could enhance knowledge through the quizzes. Every person was impressed by it, and we decided to schedule some time during the next couple of weeks to develop it.

**Aman Basnet**

In Week 2, I focused on the creation of the first backend foundation with the help of Node.js and Express. I wanted to establish the required API endpoints that our frontend would be talking with. I collaborated most closely with Apurba to figure out the nature of the data we were to get back against the user and the management of the external API calls to Google Safe Browsing and VirusTotal.

PhishTank was the first challenge that we faced. We were also going to use it as one of our sources of data, but I discovered that they were currently not receiving new subscription requests when I tried to create an account. This implied that we would have to change course and seek out solutions. I assumed the task of getting an alternative and came across OpenPhish, which seems to have a good option, not requiring an API key. My input was to produce them in plain text or as CSV, and I did this by writing a module to parse its data.

I also began writing the initial version of our /api/scan in Express. It is sent this path, and after that, it simultaneously checks with Google Safe Browsing, OpenPhish, and VirusTotal to obtain safety reports. At this point I am installing dummy responses in the form of integration testing. I organized the endpoint in such a way that I used Promise.all() so that all three scans could be performed simultaneously and shorten the latency. I also implemented error catching in the way that a failure in one service does not fail the server.

The other work that I was doing was to create the database schema in MySQL. I was generating user tables, scanning URLs, and quiz scores. Although we had not yet associated it with the frontend, it assisted us in conceptualizing how data and activities of users would be retained in the future.

**Sanjeev Oli**

During Week 2, I made my contribution in the domain of research activities and project setup. Given the fact that we were still at the point of a lot of development, my work was focused on making sure that we implemented the appropriate tools and templates to guarantee our future development. I began by supporting Hemant in the installation of Next.js to have a base to work with or to assist in the structuring of the pages we require. I also assisted in setting up Tailwind CSS to be used instead of styling and reusable components such as the header and footer.

At the time, the front-end team was busy with UI, so I undertook to research phishing detection methods. My studies were broken down into two different categories: malicious URL detection and phishing email classification. I have read scholarly articles and free-to-use tools and algorithms that examine allowed evidence based on heuristics such as length, presence of special characters, usage of IP, and so on. I presented my results to Apurba and Aman, who were able to adjust the logic we applied in our /scan endpoint.

In the case of phishing-based emails, I searched through the literature and identified some of the most common words and patterns that are present in phishing texts, including terms and phrases like :urgent, click now, and verify account. This list was to form the part of the logic in our email checker tool.

I also looked at some of the open-source datasets of phishing emails but found many of them to be out of date or not very well labeled. Accordingly, I recommended a tailor-made rule-based system to be implemented in the first version, which checks text to detect suspicious patterns. The staff liked this concept.

I was able to refine the list of features that we want to develop and also define short-term goals of each member during team meetings. I also made basic mockups in Figma of the training part where there was a quiz interface with multiple-choice questions. I mentioned the idea to the team, and we settled that it would be an effective feature to enhance user engagement.

**Week 3: Progress Report**

**Apurba Das**

Week 3 was not that much faster when it comes to development progress. I dedicated most of the time to the perfecting of API integration work that I had carried out previously. More specifically, I re-verified the request and response patterns on Google Safe Browsing and OpenPhish to make sure that the results were perceived properly. Part of the problem with the /scan endpoint was that despite its simple implementation, different situations would still affect how it responds due to the syntax required to publish to the unified endpoint, which was why it was important to test it thoroughly with several different URLs, both malicious and safe, to ensure that it would respond in a predictable way.

I also started to write helper functions to normalize data provided by the different APIs, but we did not finish with the scoring mechanism this week. Due to the limitations that VirusTotal applied on usages, I looked into a way of reducing the number of requests we would make during the development process by mocking the responses. This prevented us from going over free-tier quotas.

Other than backend work, I helped Aman to plan database fields when it comes to running URL scans. We talked about the way of storing historical scans of each user and whether we planned to log the timestamps and response metadata.

On the whole, it was a week in the shadows, a bit more than a week of actual noticeable progress. The early stabilization was done to eliminate minor self-inflicted problems and the introduction of new mechanisms. It was a little long, but we had to test the ground to make sure that we were really grounded before taking the next step.

**Hemant Adhikari**

During Week 3, I paid the most attention to the improvement of the user interface of the URL scanner form. The aforesaid input form was created in Week 2, but as far as it is concerned, it was simply made without style or feedback given. I have tried to finish the design and made it look more professional and friendly with the help of Tailwind CSS. But this week we were not able to wire it to the backend because /scan was still in the process of being refined.

Other than the minor UI work, I examined how we could show scan results of several APIs. I wrote a mock JSON object to have an idea of what we were going to receive at the backend. With this fake data, I generated a placeholder in the frontend to hold the results provided by insidious services like safety score, status, and flags by Google Safe Browsing and VirusTotal.

Optimization of route navigation with Next.js was one more minor activity that I completed. I included useRouter logic to redirect users after the event of success and began to consider an implementation of handling calls on private routes even though it was not yet completed.

In our team meeting, I raised the issue of the email phishing checker. We were not yet certain with its backend logic, and I proposed thatstick with a rule-based implementation of the first version. Nonetheless, this was not a significant component that had an implementation in the week.

The larger part of the current week was also devoted to the testing of the available frontend elements and the ensuring of the responsive design during the transition between devices. There were also some styling bugs that we found and which I fixed.

We may not have taken massive steps this week, but the small upgrades made us ready to play in the next weeks without any difficulty. Slow progress did not make the project feel like it was stalling.

**Aman Basnet**

I kept working on the /scan path on our Node.js back end. Mostly, I worked on the better error handling, mainly when one or more of the APIs, such as VirusTotal or OpenPhish, fail or report incomplete information. Because these external services are not always so reliable, I surrounded every one of the calls with try-catch logic and made sure that the server would still respond with partial results under any circumstances when required.

I even had some tasks where I experimented with mock data to practice response simulation by scanning. We were getting problems with hitting the API rate limits of VirusTotal, so it became necessary to use mock responses when testing the front-end.

A common problem I experienced was when the three APIs were not formatted in similar manners of responses. All services returned the data in different structures; hence, I created small functions that could be used to parse the data and give significant information and convert them into a structured form that could be easily understood by the frontend.

On the part of database work, very little was done in this week. The most optimal way to store user scan history is a topic that I talked to Apurba about, and we did not actually introduce anything new, therefore. Some pseudocode regarding storing results and timestamps will also be written, with the final versions to be put in place in the next few weeks.

I went through the front-end mock-ups posted by Hemant and made minor recommendations as to how we can make the results look better on the visual side. Nevertheless, this week we did not make any backend-to-frontend integration.

To put it briefly, this week was rather about tightening the screws and getting ready to work even more. Although things were not so very shiny, the work performed was significant in terms of stability and future scale.

**Sanjeev Oli**

During Week 3, I could only work with backend review and planning. I turned back to our previous work with phishing emails and started writing down the reasoning behind the detection of the phishing keywords. Nonetheless, this week I did not use the complete checker. Rather, I made a list of more than 50 common words and phrases that are usually employed in phishing, including the words and phrases verify immediately, unauthorized login, and click to update. These I was going to use later in a simple scoring method.

Development-wise, I also looked at the /scan route that Aman and Apurba had given. I tried it locally with test URLs and could see the manner in which Google Safe Browsing and OpenPhish results were delivered. I provided an input regarding the extraction of fields to be used in our scoring system, but we did not yet change that logic until the following week.

Regarding the frontend, I assisted Hemant in verifying the input of the URL, which could preview the layout of the mock results. I did not work on the code myself this week, but I was giving ideas of how the results should be presented (like a safety score meter and color-coded flags, red for suspicions and green for safe, etc.).

Nothing much was done by our team this week, with the only contributing factor being an external workload and synchronization problems. We primarily talked about the concepts and ironed out some small issues of the old arrangement. I also briefly explored publicly available datasets on phishing emails, but they did not bring anything much superior to what we have already discussed.

It looks like we did not get to make much visible progress this week, but I feel that by planning and testing this week, we are concerned that the backend logic and the frontend interface will integrate without any problems in our next development iterations.

**Week 4: Progress Report**

**Apurba Das**

During week 4, my main tasks involved completing the integration process with Google Safe Browsing in our backend. The initial connection and response format was already done, but in this coming week I will improve response parsing so as to deliver to the front only relevant responses about threats. I also implemented the logic that would deal with the situation where the URL may not give back any threat data, which ensures that the system still provides a default safe status and does not crash.

One of the minor, yet important features, was to sort our /scan endpoint. I made sure to merge the responses of the three services, i.e., Google Safe Browsing, OpenPhish, and VirusTotal, so that there is a single and clean JSON object. This schema can now be used by the frontend.

In addition to the work in the back-end, I also took part in the collective debates concerning the manner of scoring the scanned URLs. We considered a weighted approach, with the results of the API adding up to a maximum of 100 points. I was working on the formula but did not complete it as of yet.

I assisted Aman in API error management and provided feedback to Hemant on the presentation of results of the web frontend. This time I was not so much into frontend as before, so I could not make any UI contribution.

In general, Week 4 was more polishing than adding. We aimed to make sure that the backend was solid with the capacity to deal with edge cases. Although the improvements seemed minute, it was imperative to achieve quality integration and proper scoring in the future.

**Hemant Adhikari**

During Week 4, I concentrated on improving the visual display of scan results on the frontend. With the /scan endpoint becoming more stable, I used mock responses structured by Apurba to build a layout that could dynamically show the URL safety score, threat details from each API, and an overall safety status.

I implemented a simple result component that included color indicators-green for safe, red for suspicious, the numeric score. I also designed cards to separately display information from Google Safe Browsing, OpenPhish, and VirusTotal. These were only placeholders, but they helped establish how we would present scan feedback clearly and effectively.

Apart from that, I added loading and error states to the scan form so users would receive feedback while waiting for scan results. This small detail improved usability and gave our frontend a more professional feel.

I briefly discussed with Sanjeev how we might later display phishing email results in a similar format. We agreed on a plan to reuse some components to maintain consistency, but we didn’t implement that section yet.

Additionally, I helped test the backend changes by submitting various URLs through Postman and observing how the data appeared. My feedback helped identify and fix a few formatting issues in the response.

This week wasn’t heavy on new features, but it was a step forward in integrating and displaying actual results. It brought the backend and frontend one step closer to being fully connected and functional.

**Aman Basnet**

Week 4 was focused on the stability work with the back-end parts. I instead worked on optimizing the steps in our /scan endpoint, notably error handling and response schema. I introduced internal logs to aid in the debugging situation when any of the API services failed or returned unexpected data.

A major piece of my contribution this week was a prototype of a scoring algorithm. Based on the talks we had before, I wrote a function that checks the results of each of the APIs and sets a numeric value. Specifically, in an event like a URL being marked as suspicious by Google Safe Browsing, it loses a certain number of points. When all three APIs detect the URL as clean, then the score will be high. This still is very elementary in thinking, but we now have a jumping-off point.

I also worked together with Apurba to work out the final JSON output of the scan route in a more acceptably constructed format. We have included other fields such as timestamp and source breakdowns, such that the backend now produces something that is easier to read by the frontend.

I did not put too much effort into the logical parts about the database, but we have exchanged a few words about how to save scan histories of those who have entered a program. That feature is still in the planning stage.

In general, most of this week was spent on backend work, as there were no significant changes visually. The structure that we currently possess is robust enough to transition into frontend integration, more so in the upcoming weeks. I am happy with the way the pieces will take shape gradually and steadily.

**Sanjeev Oli**

During Week 4, I was working on the phishing email detection module. Additionally, even though not much code was written this week, I was able to come up with the agreed list of phishing-indicative words and phrases that we are going to use in the scoring logic. I divided them between the high-risk category and moderate-risk categories so that we can attach weighted points to each in a process of scoring the emails in the future.

I started the outline of the detection, which will have the email content as an input, and the output will be a score or a flag depending on the frequency and severity of any keyword. This logic is also at an early stage but can be easily done within a week or two.

Frontend-wise, Hemant and I wrote the template of the phishing email checker form. We agreed on a big text window where users will copy and paste the questionable content in the email and a button saying we want to check now. The result will present a score and brief explanation on why the email may be a phish. This is not yet linked with the backend.

I also had an opportunity to test the /scan endpoint perspective of a user to some extent and provided the feedback to Aman and Apurba concerning the interpretation of the scores. We talked about the inclusion of basic user feedback in the form of a text, such as a low risk or a high risk as per the numerical assignment.

Preparation of our next module was the main subject of the week. Although the phishing email checker is not yet operational, the research work and design are already in place to make it operational in the nearest future.

**Week 5: Progress Report**

**Apurba Das**

During Week 5, I worked out the uniformity of the structure of the JSON response of the /scan endpoint. The outputs of Google Safe Browsing, OpenPhish, and VirusTotal were previously being transferred as the raw responses. This week, I created more organized and clean code so that all the useful information could be processed uniformly on the frontend. This was composed of standardized threat messages, sources of threat, and indicators of safety level.

The other important task I performed was to clean up the scoring logic. Our initial model was a simple weighted model, but this week I modified it to take the different threat levels into consideration. Like with the example, when Google Safe Browsing provides a tag of MALWARE, the score will now penalize heavily. I wrote a breakdown of the response to illustrate how each source opened it up and decided which source was used and which was not used on the final score, which assists with transparency and debugging in the future.

I also got to know Hemant and started frontend-backend integration tests with him. We took dummy scan data through the API and verified whether it would render or not in the UI. Small discrepancies in how fields were named and how response after-formats were done were also corrected at this testing stage.

Finally, I checked the work on the authentication done by Aman and talked to him about how we were to limit the access of the scan history to the users who have logged in. I did not directly work on the database query this week, but I did comment on how to structure the scanned URL history by user.

This was the week that brought an important turning point as the frontend started to represent the actual backend functionality. It was gratifying to see the progress after many weeks of backend setup and optimization.

**Hemant Adhikari**

Week 5 was dedicated to making the connection between the frontend scanner form and the real backend /scan API. We had been using mock responses up until last week, but this time I applied the logic to contact real scan requests and receive real responses. This necessitated a change to the frontend API request and modifications to the parsing and display of results depending on the standardized JSON format that Apurba had introduced.

After I hooked the form, I started to polish the interface of the result display. I incorporated visual feedback according to safety scores: green color for URLs that are safe, yellow in case of moderate risk, and red in case of high risk. I also included the icons and headings of each of the data sources, Google Safe Browsing, OpenPhish, and VirusTotal, so it can be easily identified by the user what service reported the URL or none at all.

In a bid to improve user experience, I inserted a loading spinner as the scan request was being processed and an error message when the call to the API fails. These modifications enhanced the usability of the tool a great deal.

I also assisted Sanjeev in designing the first layout of the phishing email checker section. I reused the elements of the URL scanner form to maintain the consistency. Though the email logic has not been linked yet, the layout is now done.

In general, it was quite productive in terms of frontend. The real scan API insertion provided the feeling of real functionality, and the user can now paste an API and receive real-time results with visual feedback, and that was an important milestone in our project.

**Aman Basnet**

In Week 5, I focused on using the JWT-based authentication in our backend system. Another type of middleware that I wrote this week was to check JWT tokens and to secure protected routes even though we had already talked about user login and signup during the planning phase. Now the users are provided with a token on successful log-in that will have to be included in the header to get access to scan history or other user-related data.

Together with authentication, I continued with the work on scan history storage in the MySQL database. I wrote a new table to contain scanned URLs, assigned IDs of users, safety scores, and the timestamp. Once the table structure was completed, I put the first inserter version of the insert query in the /scan route to have all the scanned URLs of a logged-in user saved automatically.

I collaborated with Apurba in order to achieve the storage of only meaningful data. We cut superfluous information in API responses and stored the mandatory metadata that could be retrieved in a short period. Even though history will not load in the frontend, the backend now takes in the data properly, which pre-conditions a history view feature in the weeks to come.

I also did my part in scoring formula updates and had suggested improvements to it depending on the findings of the actual scan outputs. We have changed the weights allocated to the scores in order to balance it out more, particularly where only one of the APIs detects a URL.

In short, this week has been all about authentication, safe handling of data, and communication to the database. Such improvements on the backend are essential in making the platform personalized, secure, and future-adaptive for features of the user dashboard.

**Sanjeev Oli**

Week 5 was dedicated to the first implementation of the logic behind phishing email checking. I used the keyword lists that I had prepared in the previous weeks to come up with a function that crawls through the text entered as email and detects suspect words and phrases. The algorithm determines phishing better than others weighted by their frequency and severity, such as the words that appear frequently but with low phishing scores, like the word in the sentence, the word in the computer game, or the word that comes to mind.

This logic is now rule-based and quite basic but works on a basic prototype. I used the test of live phishing emails over the Internet and noted that the system could raise most of them with scores that were quite high. I also added a bit of logic to eliminate unwanted false alarms, like disregarding safe common statements or the ability to utilize a context check.

The layout of the email checker form was integrated on the front end along with Hemant. The user will be able to enter the existing email text into a field and then click a button to find out the probability that an email is likely to be phishing. The backend logic does not yet connect to the whole system, but the components are placed and working locally with fake results.

I also provided my feedback about the result display when testing URLs in the URL scanner, and I suggested we could use a similar system on the email checker, i.e., to indicate a possible value of phishing percentage and also to display the list of matching suspicious terms that were detected.

This week demonstrated a positive change in the email detection module. The system no longer exists as pure theory but has an operational logic layer and has an interfaced ground, bringing us further to the rollout of full features.

**Week 6: Progress Report**

**Apurba Das**

In Week 6, my focus shifted toward logging scan results and classifying them into distinct threat levels. We had already coded up the scoring logic, but this week I was able to code the results of each category to be Safe, Moderate Risk, or High Risk based on the final score. Such categorization assists the users to understand the findings better than a non-described number out of 100. But later, we decided to show both.

I further worked with Aman to help in organizing the database structure to be used to store the scan results. My objective mainly involved the necessity to make sure that every record that is stored in the database has a threat level, confidence score, and timestamp. Our purpose was to keep only the necessary data to achieve efficiency in performance.

In conjunction with this, I contributed to the resolution of minor formatting incompatibilities between backend outputs and expected frontend output. Some of these involved renaming some of the fields and flattening nested objects. I also looked at sections of the logic of result display presented by Hemant to make sure it was compatible with our backend format.

Another task that I accomplished was cleaning up and commenting on the /scan endpoint logic to allow the logic to be more maintainable for future development/documentation. I did not do any big new thing this week, although the classification system and the data logging fulfilled a vital layer of our threat assessment module.

Later this week, I started researching again machine learning models to use in email text classification and transformer-based architectures in particular. Although we are not yet adopting ML, I am currently experimenting with how it could transform or be added to our existing systems.

**Hemant Adhikari**

This week I worked on converting the response of the spam email checker into the frontend to make it easier for end users. This backend part to detect phishing email was already prepared, and I implemented logic on the frontend to display the result so that users get a phishing probability rating and their information with a confidence level, such as the term "Suspicious," "Moderately Suspicious," or "Safe," when they submit an email to be checked.

To make this interface friendlier, we introduced the use of the styled tags, which change color according to the score. I also gave a short explanation block underneath the score that shows matched suspicious keywords in support of the score. This argument section was presented to Sanjeev, and it came in handy in terms of user transparency.

I also focused on the mobile responsiveness of the scan result components. The outcomes now vertically stack on smaller scales, and all text contents get adjusted. Also, I worked on components such as the ResultCard and ScoreIndicator so that they can be reused later on within the URL and email checker interface.

In one of our team syncs, we also talked about the possibility of the inclusion of the email classification mechanism through ML in the future. Even though this is not being developed, I looked up how transformer models like bert-tiny-finetuned-sms-spam-detection might theoretically be used in our system. This is an area that we will look at deeper in the following weeks.

On the whole, this week was devoted to the enhancement of the frontend in the direction of clarity and usability, as well as to the foundation of the possible further scaling of features.

**Aman Basnet**

During Week 6, I developed and authenticated backend routes to scan data and quiz history of the user. Last week, we finally made it happen and implemented JWT-based authentication, so in the following week, I personally came up with the /history, /scan, and /email-scanner routes, as a result of which logged-in users can check their past actions. Such paths verify the token in the user and retrieve the results on the database via the corresponding user ID.

With regard to scan history, I made sure that the output contains the URL scanned, score, threat level, and time stamp. I was also involved in harmonizing my work efforts with Apurba so as to finalize the data structure that would be stored after every scan. This involved deciding on the relevant fields and schema optimization in order to retrieve the results in the shortest possible time.

I also dealt with enhancements to the logic of data insertion, where each time a scan is executed, a record has to be posted to the database automatically. I inserted fallback checks to make sure there are no empty or wrongly formatted scan data stored, which should eliminate the problem of invalid results.

I also started forward-looking on future features by prototyping endpoints to log quiz history, but the actual frontend integration still has not begun. I helped to discuss the phishing email detection using ML factors during our meetings. I searched data sets and typical approaches such as Naive Bayes and BERT text classifiers that could enhance our existing logic approach based on rules.

Even though we are still in the research stage, I have listed down the probable models of transformers and APIs that we might look into in the future. Week 6 was all backend, though the groundwork done this week will empower the majority of the personalization and history tracking in the future.

**Sanjeev Oli**

Week 6 consisted of the optimization of the phishing email checker logic and explanations of outputs. I went back to the keyword scoring system, and I changed weights according to severity. Classic words like "urgent," "account update," or "security alert" can now become a greater contributor to the final score of being phished. Such an update provides a more realistic adherence to the risk of phishing.

I have also introduced new behavior in the backend response, a reasoning field. This section of the output reveals to users the keywords that motivated the score and on what grounds. The argumentation form is small in size, with the keywords as well as a brief explanation being mentioned. To use an example, in case of a finding such as “verify your account,” they append a reason such as “Uses common account verification bait.”

Once the backend logic got updated, I partnered with Hemant to begin frontend testing. We experimented with the presentation of the matched keywords and reasoning that can be simple and adjusted the presentation in a better way. Such cooperation meant that users merely did not obtain a number but had an idea of why the email was flagged.

I also had started to do some early research on machine learning techniques of spam detection early on. I went through a few open-source models like roshana1s/spam-message-classifier and read about their architecture, the size of the dataset, and the complexity of the integration of the same. We haven’t tired of integrating ML now, but this experiment will enable us to determine whether a hybrid rule-based and ML approach could give more promising results.

In general, the main aim of Week 6 was to increase the informative, user-friendly, and technically sound aspect of the email checker. We came a long way in reconciling backend logic and face-level explanations when designing future upgrades that incorporate ML technologies.

**Week 7: Progress Report**

**Apurba Das**

During Week 7, I was working on the design of the backend logic that would have the user dashboard feature. This included the process of how we would be retrieving the historical scan information depending on the authenticated user. I designed the framework of the query that retrieves URL scans in descending order of date along with being classified according to threat level. As well, I collaborated with Aman to test and fix the backend routes that retrieve these history records safely with JWT authentication.

The format and organization of the scan history output is another one of the tasks that I took care of this week. I made sure that fields such as scanned URL, score, threat level, and scan timestamp were well explained in the backend response. This simplifies the way Hemant can process the data and present the data on the frontend dashboard in a clean way.

Also, I have been involved in studies about the possibility of the use of ML models in our phishing tracker applications. I read through some of the more common transformer-based models, such as bert-tiny-finetuned-sms-spam-detection and roshana1s/spam-message-classifier, to inform myself on how they were designed and their data requirements. Although we have not started any implementation, this research has provided an insight into the possible integration practice in the future.

During the week I was also involved in the process of debugging some backend inconsistencies present during history retrieval and classification filtering. These are small changes that had big implications, as they made data more consistent on the platform.

Week 7 was constructive in making a shift in our attention towards user-specific features of the core scanning logic. Now that we have both our backend dashboard logic and ML research well on the way, we are gearing it up as we prepare to finally integrate and polish in the coming weeks.

**Hemant Adhikari**

The major work that was done during Week 7 was designing and developing the user dashboard layout. The plan was to enable the user to have an overview of their scan records and quiz performance in terms of progress through a responsive and well-structured interface. I started with the dashboard design, wireframing the layout into categories, or rather dividing it into such parts as Scan History, Quiz History, and Account Info. Once I approved the design, I began developing the dashboard in Next.js and Tailwind CSS.

In regard to the Scan History section, I made a React-based table component that would list all previous scans of links, their score, threat level, and date but later I figured out cards would be better. I collaborated with Apurba and Aman to match the frontend structure to the backend API response. After finishing the API route and testing it, I retrieved and displayed genuine user data in the dashboard, which is authenticated using JWT.

Meanwhile, I helped to plan the Quiz History section. The quiz tracker was not functioning optimally yet; still, I made some dummy components and configured the framework for future inclusion.

Our team discussion also involved how feasible it would be to incorporate ML models within the pipeline of phishing detection. I also assisted Sanjeev in looking at UI requirements and how we could eventually display ML results alongside our rule-based results to allow comparison. Though this remains theoretical, we are seriously contemplating this feature as a future upgrade.

Finally, I added a couple of tiny UI fixes and resolved some bottlenecks with the responsiveness on smaller resolutions. The dashboard format has been optimized to work across the devices.

**Aman Basnet**

I worked on backend development of the user dashboard this week. I completed and fixed the path to retrieve user-specific scan history and quiz history. The /api/history endpoint should be able to return every scan done by a logged-in user with URL, threat score, threat level, and timestamp. The responses are filtered and ordered by time, and further measures were implemented to avoid unauthorized access.

I worked with Apurba in a manner that made the data structure conform to frontend specifications. I added query-level filtering and indexing on the database in order to maximize performance. This will enable quicker access as the number of scans per user goes higher.

I started to write the logic of the Quiz History endpoint as well. Although the quiz functionality is not complete to date, I laid the groundwork for where the quiz results will be stored when the feature is launched online. It is now in a position to store the date, score, and question set used against each quiz.

Both in relation to backend routes and my personal research, I have studied machine learning-based phishing detection. I practiced with dataset formats that can be utilized in spam detection problems and examined model interfaces on Hugging Face. I aimed at recognizing the models, which could be implemented in the framework of our project and operate in the available resources. I checked how it is possible to fine-tune such a model as BERT-tiny using it to detect spam and how it is possible to develop JSON-based APIs to serve predictions.

As we are in research mode, we put in the groundwork this week on both backend functionality we need now as well as future upgrades based on ML technologies.

**Sanjeev Oli**

My main task in Week 7 was machine learning models related to spam email detection research and how we could add one into a platform. To start with, I searched through a number of free spam detection models on Hugging Face. The reviewed include bert-tiny-finetuned-sms-spam-detection, roshana1s/spam-message-classifier, and knkarthick/bert-base-cased-finetuned-cola. I read about their input format, training data, and prediction formats.

The concept would be to understand the manner in which these models classify and whether these models could work on an intermediate layer to our current rule-based system. I wrote down their advantages, disadvantages, size of their models, and the easiness of their integration. Another issue that was brought up in our team was the possibility that inference on the server was too resource-intensive without a GPU, so I also considered light models.

Working on the implementation side, I assisted Hemant with formatting a potential ML-based output, which would display beside our current phishing score. The UI was mocked to have both results presented one above the other to enable the user to comprehend the disparity and increase a level of trust in our system.

Other than ML work, I was also optimizing the phishing email checker. I incorporated logic to niche and emphasize the most effective matching keywords. These are now in bold in the reasoning field, and the confidence label is therefore changed.

I also tested how my backend will be integrated with Hemant's frontend work and also checked the updated phishing score output. Week 7 played the key role in connecting our current rule-based system to our future ML aspirations to enhance the explainability of our results today.

**Week 8: Progress Report**

**Apurba Das**

Focusing on finalizing and eventually incorporating our phishing detection model into the backend was the work I was mostly engaged in during Week 8. Several weeks into the investigation and experimentation, we were ready to replace our keyword-matching framework with a Hugging Face transformer model: ealvaradob/bert-finetuned-phishing. This model worked well on classifying phishing and non-phishing emails and gave us confidence scores, which was easy to incorporate into our current scoring logic. Using this confidence score, we were able to dynamically display threat levels without dense manual rule systems.

I also refactored the /email-scan endpoint to support this model and also ensured the classification label, confidence, and threat summary were stored in each scan. I also did some work on optimizing this endpoint so that it can better deal with Hugging Face APIs lagging or throwing errors and properly give a user the necessary feedback when this happens.

At the same time, I assisted the frontend team in the metadata formatting and uniform response shaping of the new model. A common decision on the functionality of a quiz was also made with my contribution. We did not store the quiz score of the user in the database since the quiz will only serve to train user awareness and not as a metric of assessment. This lessened data overhead and made the backend routes simple.

Lastly, I worked with the team on the optimization of the main landing page. We also refreshed the information to be more descriptive of what we offer and included graphics to increase user confidence. Our tools and the new ML-driven features are now brought to the forefront of the page.

All in all, this week was a giant leap in feature completeness and clarity of user-facing explanations of phishing threats.

**Hemant Adhikari**

The activities that I had this week fully engaged me in the work of the team and included the implementation and refinements of the three following areas: the email checker interface with the Hugging Face ML model, the refinements of the landing page, and UI changes depending on the model confidence scores.

Now with the new model (ealvaradob/bert-finetuned-phishing) integrated by Apurba, I modified the frontend logic of the email scanner to obtain the prediction label and confidence score. With such a high confidence score, it was quite simple to visually and intuitively classify it into categories like "High Risk," "Moderate Risk," and "Safe." It also enhanced the comprehension of the user on why a message had been flagged.

Where we stripped out the text in the earlier explanations of keywords by rule, I changed the explanation in the second section to indicate model-based results. At this point the UI shows the label, confidence (e.g., 92.4%), and a summary message. I ensured it was responsive to every screen size.

On the quiz part, as a team we even discussed internally and decided that we were not going to track or store quiz scores in the backend. Saving scores was not significant since the aim is to increase cybersecurity awareness; it would interfere with the learning process. I wrote out any database dependencies of the quiz components and left it entirely client-side.

Lastly, one of the areas in which I worked was on the main landing page. I restructured its design in order to highlight such features as URL scanning, phishing email detection, and training quizzes. I also included call-to-action buttons and made it more simplified in order to fit the guidelines of minimalistic UI. The alterations contributed to the creation of a more coherent atmosphere on our platform.

**Aman Basnet**

I worked on backend modifications in Week 8, which were required after we switched to the Hugging Face transformer model for phishing email detection. I worked with Apurba to refactor the /email-scan endpoint in order to incorporate ealvaradob/bert-finetuned-phishing. We are able to abandon our old rule-based method that relied on matching keyword patterns because the new model provides a clear label for the categorization and confidence score.

The value of confidence created by the model was particularly useful. I changed the way of scores to conclude the overall value by using the value of the score, thereby minimizing manual adjustments of weights and logic of phishing occurrence. My other provision was to make sure the scan records were still saved successfully to the database and there was no alteration in the structure of data.

There was also a team decision to disregard the idea of storing quiz scores in the database. As it is not so much an assessment to be completed but an interactive awareness module, making the quiz stateless was a more sensible thing. I eliminated the endpoints and schema drafts of quiz history, which simplified our source code.

Also, I was instrumental in testing the entire system integrity following the appearance of the new model. I confirmed that the ability to retrieve a scan history remains intact, that the machine learning–based results are correctly read and returned, and that the API still operates effectively without exceeding timeouts or encountering inconsistencies.

Towards the final days of this week, I assisted the frontend team to align new forms of scan results. We are now near complete feature parity, have fewer bugs, and are more intelligent in email classification.

**Sanjeev Oli**

A significant change in the phishing email detection system was achieved during Week 8. I collaborated with the team to switch the previous rule-based approach to a more intelligent model based on transformers by the much smarter and more reliable successor Hugging Face model ealvaradob/bert-finetuned-phishing. This model enabled us to categorize email into phishing and safe, and it gave a confidence score. Not only has it made our detection system more accurate, but it has also made scoring and UI explanations a lot easier.

By getting rid of the keyword-based system, I cleaned up the backend logic that took care of keyword highlighting as well as severity weights. I also changed the logic applied in checking the presence of a threat and assisted in formatting the backend response such that it would be easy to understand and handle on the frontend.

At the same time, I consulted Hemant on how to reflect this change in the email checker UI. We also removed keyword lists and got model-based summaries, and now the interface is centered on the confidence score and classification label. That seems to make the tool more professional and advanced.

I also agreed with the move to leave the quiz feature to be frontend only. The purpose of a training tool was discussed, and we decided that storing scores would serve no useful purpose but just to complicate things. I did some small changes on the content in the quiz section to aid in clarity and readability.

Lastly, in order to further showcase our capabilities-especially with our new ML-based email detection-I assisted in rewriting parts of the landing page code. With the new landing page clearly communicating our tools and goals, the cybersecurity awareness mission has been accomplished.

**Week 9: Progress Report**

**Apurba Das**

During the last week, I worked on making the backend services stable and reviewed them. I returned to the /scan and /email-scan endpoints and checked behavior on error and response regularity and structure. Slight anomalies—such as empty fields or label omissions in edge cases-were fixed.

I was also working on technical documentation of our backend services. This involved an endpoint description, sample requests and responses, and JWT flow authentication. The documentation is also formulated in a manner that whoever develops or reviews our system in the future could easily comprehend the logic and ability of our system.

The other thing was summarizing our findings on the machine learning research. In the previous weeks, we became familiar with models such as ealvaradob/bert-finetuned-phishing, and in Week 9, we presented the practical results. It is reliable, very lightweight for inference API-based, and it cleans up well within our Node.js backend. These results were listed, and they formed part of the final report.

I additionally arranged end testing for all major routes and functions to make the system reliable across the board. This involved the entire QA round /history route, with both the URL scan records and email scan records getting stored and retrieved properly.

Week 9 involved closing of the files and getting them stable and accurate. In retrospect, I would say that I am impressed by the development of the scanner into one of the most complete cybersecurity awareness programs supported by machine learning.

**Hemant Adhikari**

On Week 9, I focused on the last round of polish to the frontend interface, testing of responsiveness, and preparation of the platform to demonstrate. The accessibility tests that I have done on various devices and screen sizes are to be sure that our platform is consistent among users. I solved minor UI bugs within lower screen resolutions and changed font size and padding in between parts to be easier to read.

I also did a test on all the featues - the URL scanner, phishing email checker, and taking a quiz. This was to make sure that everything was working as expected with a live backend reply. Particular attention was paid to the fact that the results of the email scan done by ML were consistent and the confidence score was shown in the appropriate format.

One of the key duties in which I participated was documentation. I also created front-end usage guides, commented on code snippets, and assisted in the final user documentation to be submitted. These were guidelines on how the users can make use of our tools as well as interpret the results of scans and the descriptions of the probability of being a phisher.

Further, I collaborated with Sanjeev to provide finishing touches to the several quiz sections. Scores are not stored, but we made sure that the quiz would look and feel highly interactive and dynamic in terms of visual feedback and transitions.

I then assisted with the formatting of the landing page and a last look into the messages. We focused on being clear, easy to understand, and trustworthy, and therefore, the users can recognize the platform is designed to be aware and safe.

In general, Week 9 was the summary of everything. The platform currently works fine, is clean, and is to be made ready to be presented and demonstrated.

**Aman Basnet**

During the last week, I did a full test and security updates on our backend services. I have tried all possible route edge cases - e.g., missing fields, invalid tokens, invalid requests, etc. and made sure that the API is predictable and results in meaningful error messages.

I added the rate limiting and a security middleware as well to prevent endpoint abuse. This came with restrictions on the number of scans a user may perform during a short period of time and input validation on requests to eliminate suspicious input formats. They were necessary late requirements in order to strengthen our system and become more production-oriented.

Documentation of the overall architecture of the backend was one of the major contributions that I made this week. I made a visual illustration and description of how every element is interlinked together - route handlers, middleware, external APIs such as Google Safe Browsing, and the Hugging Face model.

Regarding machine learning studies, I assisted in the conclusion that, instead of training our own models, we should employ Hugging Face APIs. I included cost, performance, and ease of integration notes in order to justify this design choice. This is the reason we incorporated this rationale into our final deliverables.

The scan history as well was re-tested, and I ensured that the email and the URLs were being logged adequately and allowing retrieval of the same by authorized users.

In general, Week 9 was cleanup and verification and packaging. Our backend is steady, robust, and documented. I felt good to view the entire system working properly and how our initial concepts were translated into a whole product.

**Sanjeev Oli**

During Week 9, we worked on the final testing and user experience touch-ups and collected results of the machine learning vs. rule-based phishing detection in our report. Once having a Hugging Face model perfectly incorporated into the email checker, I performed several test cases of both spam and legit emails to check its precision and dependability. The findings were reliable, and the project followed a large confidence and easy incorporation into our front end.

Assisting in the preparation of final evaluation material is one of the main contributions made this week. I prepared a couple of phishing and non-phishing mails that were used as samples to demonstrate the functionality of the system. These were assembled as a set of examples of areas where the ML model is strong, like in identifying phishing attempts that do not contain obvious keywords.

I have also documented the migration out of the original rule-based keyword matcher to the modern ML-based model. I made a small table of comparison of the differences in terms of flexibility, accuracy, and confidence scoring. We have added this table to our final project report to see the technical shift of the email detector module.

I, together with Hemant, made sure that the quiz module was polished. Even though there is no storage of scores, visual feedback, transitions, and verification of answers have been refined. I also checked all the quiz questions to see whether they are easily understood or educational enough to be useful in giving a new form of awareness.

Finally, I tested the entirety of the live site and reported any bugs/inconsistencies and wrote my final comments to the team. Week 9 wrapped up in a working, well-tested system that can detect threats and also teaches users-a good way to end our cybersecurity project.