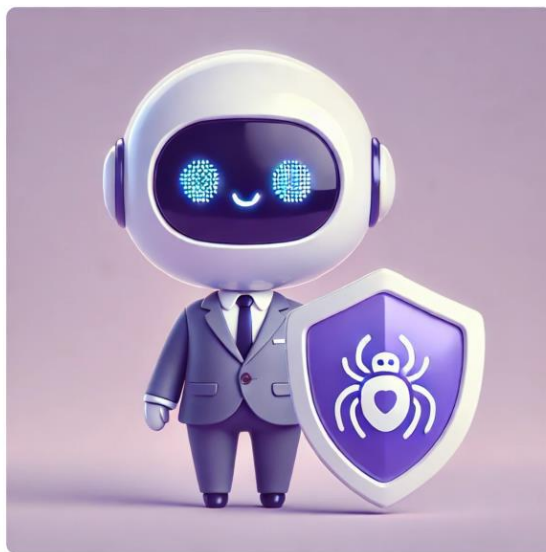


COS70008 – Technology Innovation Project and Research

Final deliverable

Project Reflection Report

CyberShield AI – Hybrid AI System for Malware Detection and Classification



Cyber Shield AI

Know the Enemy. Outsmart the Threat.

Register

Login

Student Name: Arun Ragavendhar Arunachalam Palaniyappan

Student ID: 104837257

Date: 2 /06/ 2025

Project Title: CyberShield AI – Hybrid AI System for Malware Detection and Classification

Table of Contents

1. Introduction	4
2. Group Work Reflection	4
2.1 Strategies and processes that worked	4
2.2 Strategies and processes that did not work	7
2.3 Things that could be improved next time for a group work	10
2.4 Outstanding Events and Contributions	10
3. Individual Work Reflection	13
3.1 Individual Tasks done in Each Project Phase	13
3.2 Individual Contributions to the Group as a whole	22
4. Conclusion and Recommendations	23
4.1 Conclusion	23
4.2 Recommendations for Further Development	24
5. References	25

List of figures and tables

- Fig 1: Tien creates the MS team group*
- Fig 2: Vatan started a poll to decide team meeting times*
- Fig 3: Vatan started a poll to decide team meeting times*
- Fig 4: Trello board created by Arun*
- Fig 4: Design concept report submitted and Arun appreciated by team members*
- Fig 5: Design concept report submitted and Arun appreciated by team members*
- Fig 6: GitHub Repository created by Arun*
- Fig 7: GitHub Repository created by Vatana*
- Fig 8: Cluttered and messy whiteboard with all designs on a single page*
- Fig 9: Nicha messaging that she cannot come to the meeting at the scheduled time*
- Fig 10: Vatana helping Tien with the system Integration of Design 3*
- Fig 11: Arun informing the team that Naveed has approved Design 3*
- Fig 12: A meeting room booked for the team by Arun*
- Fig 13: Meeting minutes posted in Canvas group*
- Fig 14: Sharing preliminary design and technical tips*
- Fig 15: Meeting discussion after submitting Design concept report*
- Fig 16: Meeting discussion after submitting the presentation*
- Fig 17: Team gathered and rehearsed the whole presentation and demonstration*
- Fig 18: THE CIC-MalMem2022 dataset downloaded from Kaggle [7].*
- Fig 19: CyberShield AI System Architecture [1,2,3,5,6,]*
- Fig 20: Design 1 Figma*
- Fig 21: Autoencoder [3,5]*
- Fig 22: Random Forest Classifier [1,6].*
- Fig 23: Model getting Trained*
- Fig 24: Model finished training*
- Fig 25: Predictive Risk Rating Module [5]*
- Fig 26: Full Backend code directory from main project directory*
- Fig 27: Flask server successfully running*
- Fig 28: Full Frontend code directory from main project directory*
- Fig 29: Front end page where user/admin can scan a file*
- Fig 30: Ransomware malware found*
- Fig 31: Spyware malware found*
- Fig 32: Trojan malware found*
- Fig 33: Analytics dashboard*
- Fig 34: Analytics dashboard*
- Fig 35: Admin retrains the model successfully [4]*
- Fig 36: Technical support for Vatana (Design 2 – Model)*
- Fig 37: Technical support for Tien (Design 3 – routing)*

1. Introduction

This report provides a detailed reflection on the group and individual work done during the 12-week COS70008 Technology Innovation Project. The main goal was to build a three-tier web application that detects, predicts, classifies and analyses malicious attacks using a hybrid AI model.

Five system designs were proposed to the client who chose three for full stack implementation.

The report includes:

Group experiences (Weeks 2–11):

- Setting up team communication and weekly meeting slots.
- Assigning full system designs tasks to each member.
- Creating and maintaining shared documents (Gantt chart, Trello board).
- Collaborating on Figma prototypes, backend APIs, and testing plans.
- Compiling and submitting the Innovation Concept design as a single report.
- Preparing and presenting a live demo for the client.

Individual contributions (Weeks 1–12):

- Understood the project brief and clarified technical objectives.
- Researched hybrid AI models to address an individual learning gap.
- Worked on end-to-end development of Design 1 - CyberShield AI, including:
 - System architecture and ML design.
 - Figma UI and Vue.js frontend.
 - Dataset preprocessing and model training.
 - Flask backend and SQLAlchemy database.
 - Testing, final presentation, and client delivery.

2. Group Work Reflection

2.1 Strategies and processes that worked

Team communication and weekly meeting setup

- The team used Microsoft Teams for all planning. In week 2, Huu Tien Dang created the group.
- Vatana Chhorn ran a poll, and Thursdays 12:30 – 4:30 PM and Saturday evenings were fixed as the weekly team meeting slots.
- Arun (myself) led the initial brainstorming session to align everyone with the project goals and client expectations.

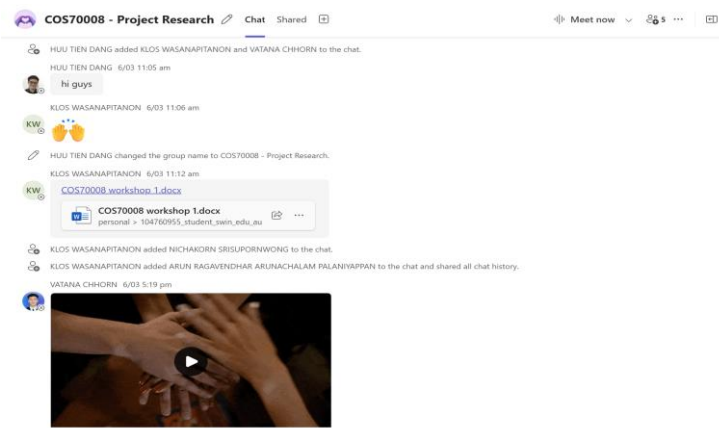


Fig 1: Tien created the MS team group

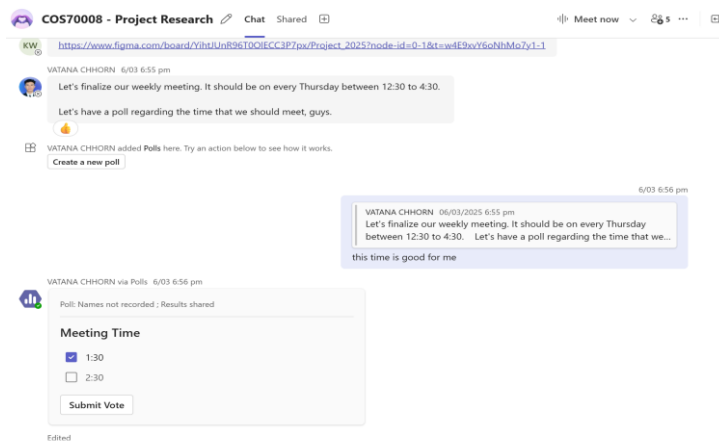


Fig 2: Vatan started a poll to decide team meeting times

Assigning one full design to each member

- Instead of dividing work by roles (which initially caused issues), each member was given full responsibility for one system design.
- Arun developed Design 1 (CyberShield AI), Vatana handled Design 2, Klos worked on Design 3, Tien took Design 4, and Nichakorn managed Design 5.
- This approach worked well, allowing members to focus deeply, avoid cross-dependencies, and move at their own pace.

Name	Figma Prototype	Implementation Group	Main Responsibilities	Individual Learning Focus
Arun (Myself)	Prototype 1	Implementation 1	Team leader, AI Model development, Full stack web app development, AI model integration for malware detection	Hybrid Machine Learning for - Malware classification and behavior analysis in CPS
Vatana	Prototype 2	Implementation 2	Hybrid ML model development and tuning	Phishing - Classification hybrid machine learning algorithms and techniques
Tien	Prototype 3	Implementation 2	Data preprocessing, pipeline integration, model input setup	Data preprocessing for machine learning
Klos	Prototype 4	Implementation 3	UI/UX design, dashboard and visual output	Interface and data visualization dashboard
Nicha	Prototype 5	Implementation 3	Frontend (ReactJS), testing, data handling, basic security setup	Data security framework

Fig 3: Vatan started a poll to decide team meeting times

Shared task tracking and file management (Trello board, OneDrive)

- Arun (myself) created a detailed Trello board with weekly task cards for each design. Labels covered Figma, ML, backend, testing, and demo.
- Everyone updated their tasks regularly, helping the team stay on track.
- OneDrive was used to store and co-edit Figma files, reports, and presentation slides, keeping everything organised and accessible.

WEEK 1-2	WEEK 3	WEEK 4	WEEK 5	WEEK 6
Group Formation Introduction to the project	Individual Literature review on whole project Identifying learning gaps	Deeper, specific Technical review Listing out the features for 5 unique design solutions Searching for well detailed datasets	Narrowing down on the technical research Finalizing the 5 designs Start Wire framing and developing the Figma prototypes	Finalizing and refining the Figma prototypes Presenting it to the client
WEEK 7	WEEK 8-9	WEEK 10	WEEK 11	WEEK 12
Start the development of the 3 designs based on the client's choices Cleaning, pre-processing and splitting of collected datasets for AI model training	Hybrid AI model training - 3 different and unique design combinations	Three tier Web App Development with database, user interface and back-end logic	Integration of the Web apps with the hybrid AI models Final testing and debugging of the full stack system	Project Delivery and presentation to the client

Fig 4: Trello board created by Arun

Innovation Concept report compilation and submission

- In Week 6, Arun (myself) took the lead in compiling the Innovation Concept Design Report.
- After collecting individual write-ups, Arun cleaned, rewrote, and formatted the content into a single consistent document.
- The draft was reviewed by all members on Teams and then submitted to Canvas.
- Arun (myself) received appreciation from team mates for taking the extra effort.

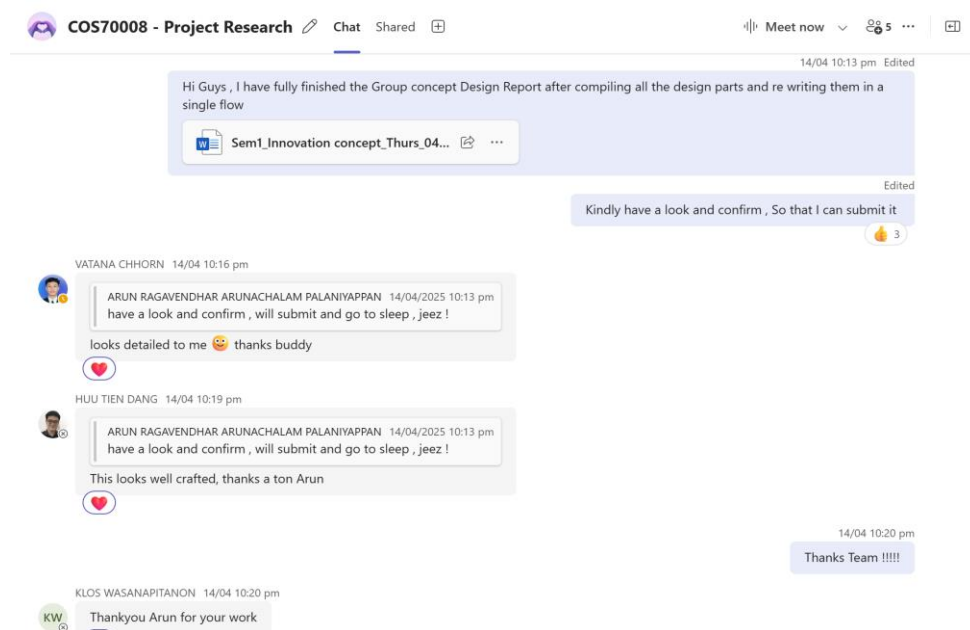


Fig 5: Design concept report submitted and Arun appreciated by team members

Creating and Maintaining Git Hub repositories

- Arun(myself) and Vatana created separate GitHub repositories for each design.
- They helped standardise the folder structure and naming conventions across all repos, making the codebase easier to review and manage.

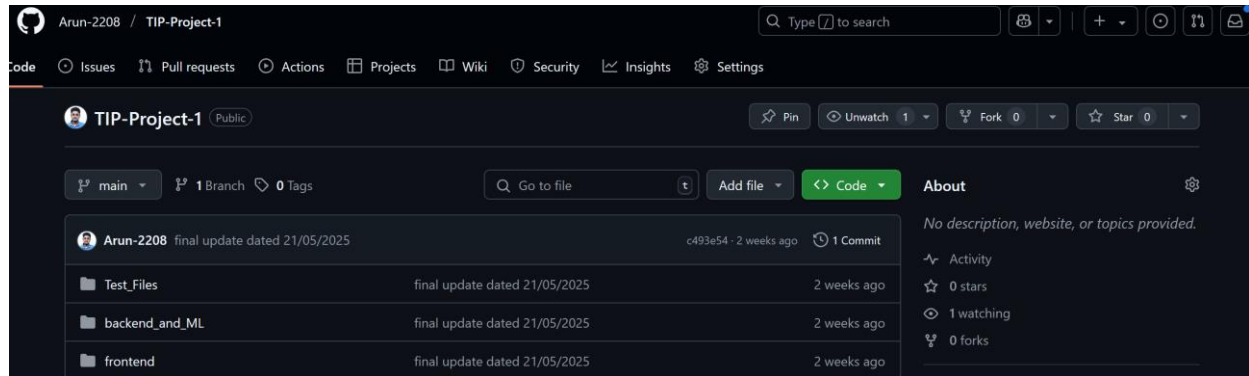


Fig 6: GitHub Repository created by Arun

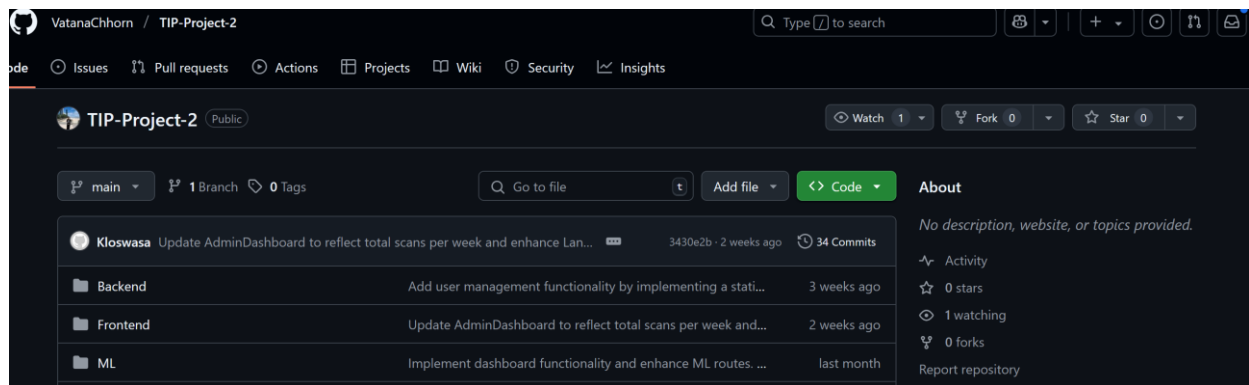


Fig 7: GitHub Repository created by Vatana

2.2 Strategies and processes that did not work

Initial technical skills-based task split caused delays

- Work was first divided based on strengths—Arun on machine learning, Klos on frontend, and Vatan on backend.
- This led to confusion and delays when tasks depended on each other and would have made final integration difficult.
- The team resolved this by assigning one complete system design to each member.

Cluttered shared board made coordination harder

- All five designs were placed on a single whiteboard early on. It looked messy and hard to follow, making development confusing and slow.
- Each design should have had its own page, which would have made both coordination and demonstration easier.

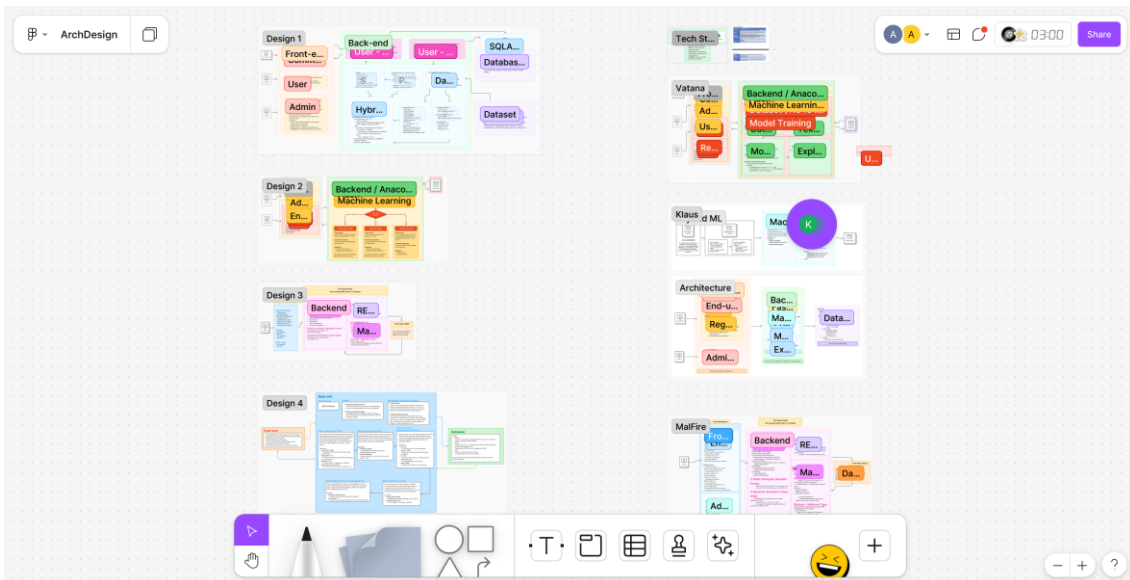


Fig 8: Cluttered and messy whiteboard with all designs on a single page

Some team members could not properly attend weekend meetings

- Regular online meetings were scheduled for Saturday evenings, as agreed earlier. However, Tien and Nicha had part-time jobs and often could not attend.
- This created inconvenience, especially as Arun (myself) and Vatana were free during that time and preferred fixed weekend slots. This affected team sync in Weeks 7-10.

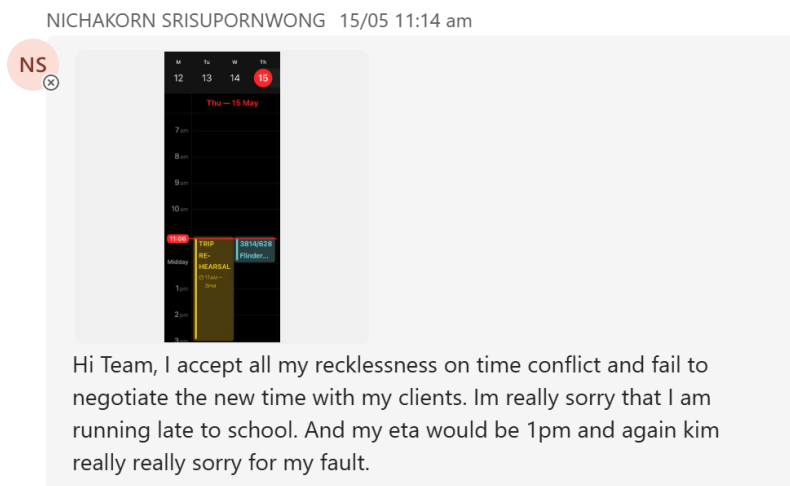


Fig 9: Nicha messaging that she cannot come to the meeting at the scheduled time

Frontend-backend integration started very late for Design 3

- Design 3's backend was completed by Week 9 (Tien), but frontend work began only in Week 10 (Nicha).
- This left very little time for integration and polish, especially for PDF generation, charts, and risk scoring.
- Vatana stepped in to support integration during the final week, to help with the rush.

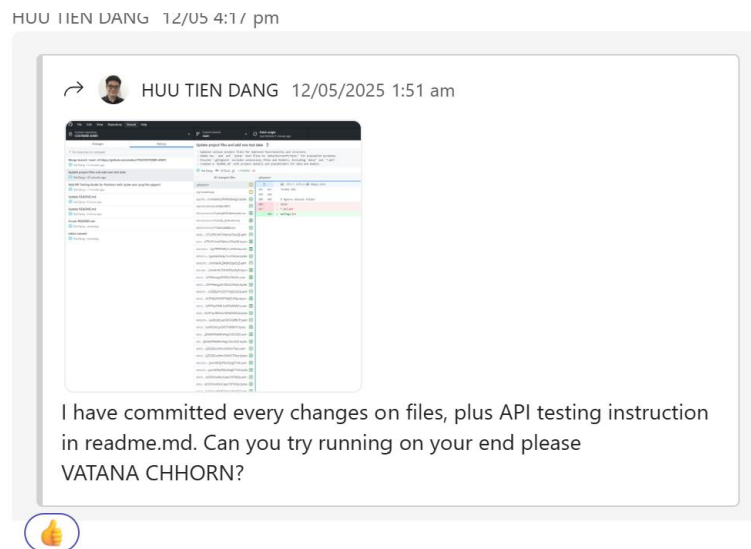


Fig 10: Vatana helping Tien with the system Integration of Design 3

Limited cross-checking across designs in early weeks

- Only Arun (myself) and Klos regularly reviewed each other's designs for alignment with the client brief.
- This was not done by all members. Earlier group-wide reviews could have improved logic and consistency across all five systems.

No mid-project quality review

- There was no formal review around Weeks 7–9 to assess progress or design quality. This allowed small issues to build up.
- For instance, Design 3 faced delays because Tien and Nicha had not finalised their design direction.
- Arun stepped in to finalise Design 3 and get it checked and approved from the tutor.
- A structured check-in could have helped flag such gaps earlier.

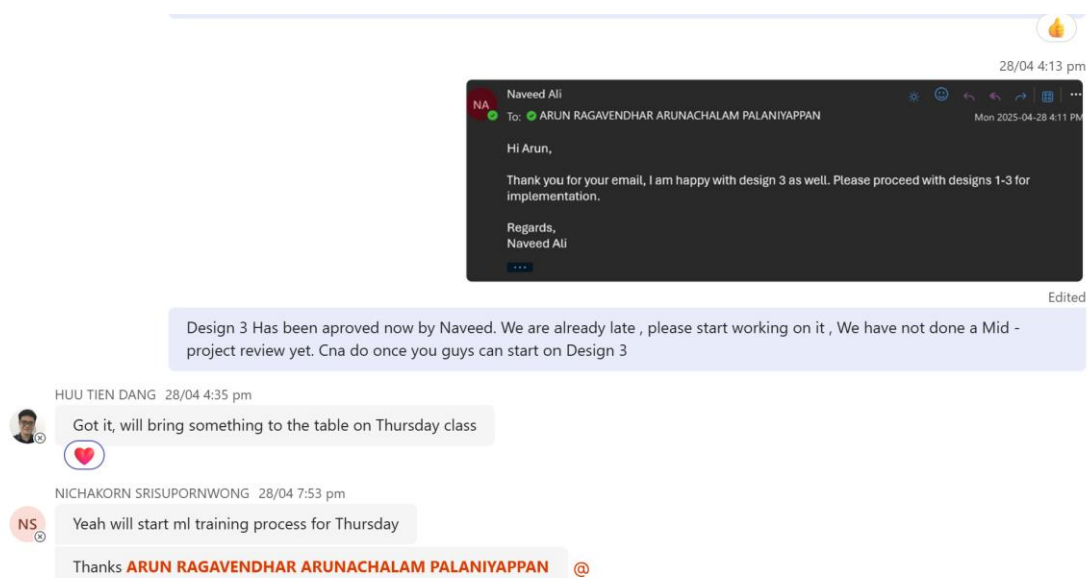


Fig 11: Arun informing the team that Naveed has approved Design 3

2.3 Things that could be improved next time for a group work

From my experience during this project, the following improvements would make future group work smoother and more efficient:

- **Starting frontend-backend integration earlier:** In Design 3, backend development was done by Week 9, but the frontend started in Week 10, leaving little time for full integration and UI testing. A more parallel approach from the beginning would avoid last-minute pressure. A similar issue happened in Design 2 as well.
- **Scheduling a formal mid-project review:** There was no structured review in Weeks 5–7, which allowed some delays and design gaps (especially in Design 3) to go unnoticed. A proper check-in could have helped align everyone, flag unclear designs, and fix issues early.
- **Appointing a rotating weekly reviewer:** Having one team member each week check all designs for consistency, logic, and alignment with the client brief would help maintain overall quality. This would also spread responsibility fairly and promote collaboration across designs.
- **Ensuring cross-checking is team-wide, not just between two members:** Klos and Arun (myself) regularly reviewed each other's work to ensure it matched the client brief. If all team members had done this, the overall quality and consistency would have improved even more.
- **Setting a more inclusive meeting time:** Saturday evening meetings did not work for some members with part-time jobs (Tien and Nicha), which made team coordination harder in the middle weeks. Choosing a slot that fits everyone's availability early on would improve attendance and communication.

2.4 Outstanding Events and Contributions

Team Organisation

- Arun (myself) created a detailed Trello board with weekly task cards for each design. This helped the team visualise workloads and stay on track.
- As team leader, Arun (myself) also shared key updates regularly, clarified doubts, and supported teammates when needed, strengthening collaboration and maintaining team focus.

Meetings

- Klos, Tien, and Arun(myself) consistently booked group rooms at Swinburne Library for focused group sessions. For example, Room 4 was used on April 6 to finalise Figma flows, API designs, and dashboard layout.

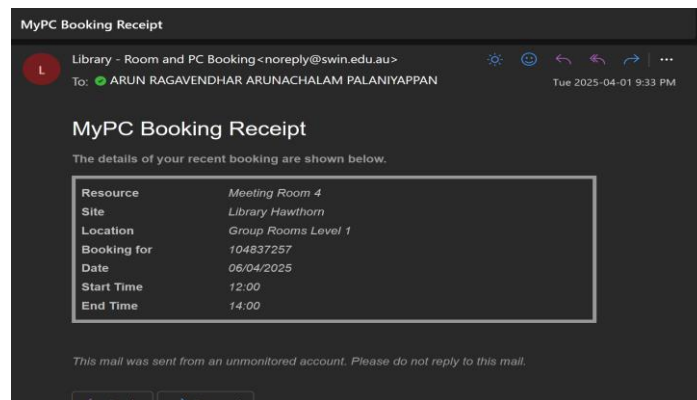


Fig 12: A meeting room booked for the team by Arun

- Arun(myself) also posted meeting minutes to Canvas to keep everyone informed, including those who could not attend. These sessions were well-organised and improved team alignment.

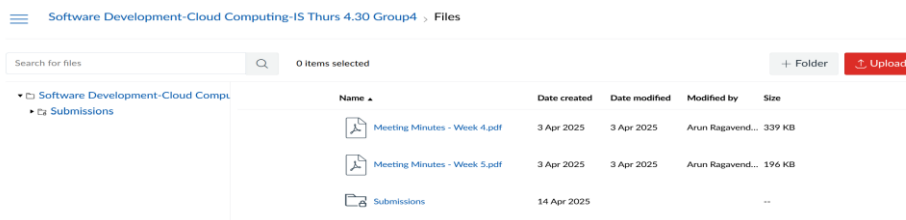


Fig 13: Meeting minutes posted in Canvas group

Delivery of Project Design Ideas

- On April 4, Arun(myself) shared the architecture flow for the preliminary design in Teams. It explained how data moved through the hybrid AI model to risk scoring.
- This helped teammates clarify their own logic and maintain design consistency, a clear example of cross-team support.

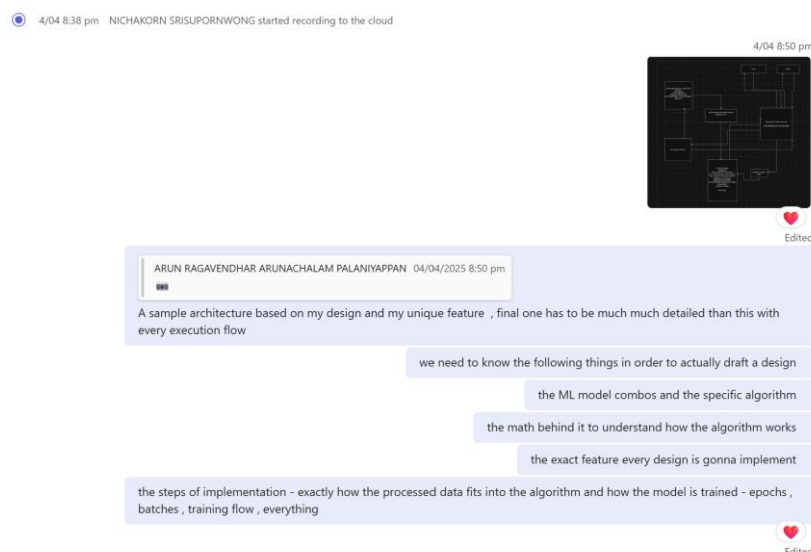


Fig 14: Sharing preliminary design and technical tips

Innovation Concept Report Compilation

- By Week 6, all team members submitted their design sections. Arun took the lead in compiling the full report, refining content, aligning visuals, and ensuring formatting met the rubric.
- The draft was reviewed on Teams, with positive feedback from Tien, Vatana, and Klos. It was a well-coordinated effort and a clear example of strong group collaboration.

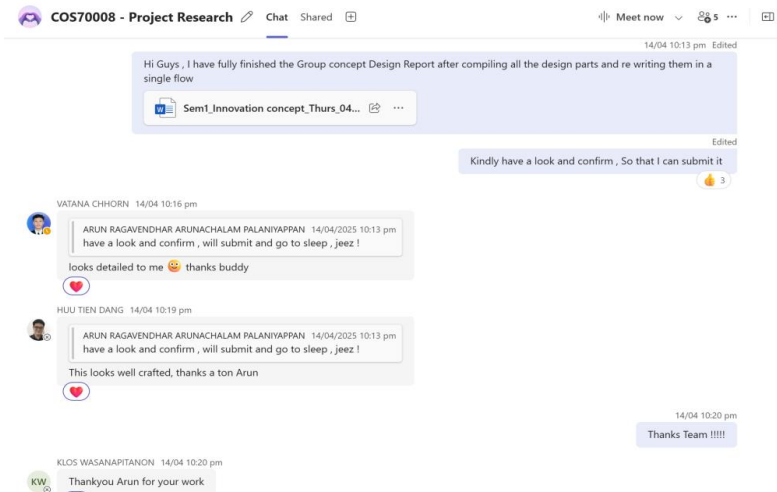


Fig 15: Meeting discussion after submitting Design concept report

Final Client Presentation

- In Week 12, team prepared the final presentation and led the overall demo structure. The team rehearsed multiple times, wrote clear scripts, and refined transitions together.
- All team members clearly explained all three designs, while also answering client questions, about all designs.
- The presentation was smooth, well-coordinated, and reflected the team's shared understanding and preparation.

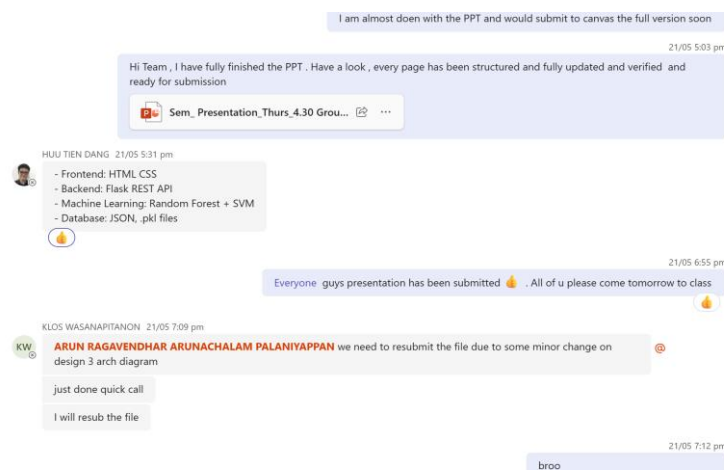


Fig 16: Meeting discussion after submitting the presentation

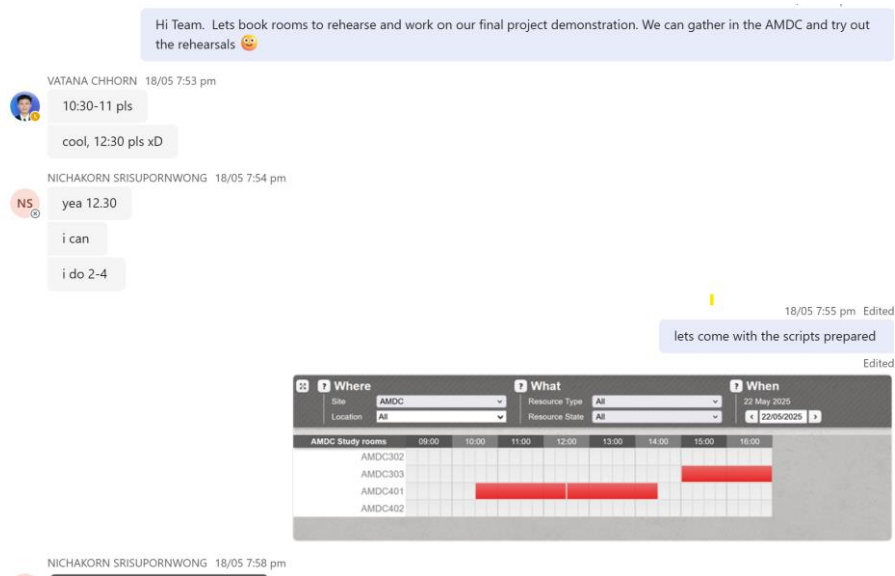


Fig 17: Team gathered and rehearsed the whole presentation and demonstration

3. Individual Work Reflection

The primary individual contribution was the complete design and full implementation of Design 1 – CyberShield AI.

Full Project's Repository: <https://github.com/Arun-2208/TIP-Project-1>

Link to Figma: <https://www.figma.com/design/LeODbt9l38BV7VTPap9Sub/CyberShield-AI?node-id=125-3496&p=f&t=gZKmVH2bDgBXLgeS-0>

3.1 Individual Tasks done in Each Project Phase

Phase 1

(Weeks 1–3) - Planning, Goal Setting, and Technical Research

- The project brief and client requirement were carefully studied to find learning gaps.
- The learning issue was about understanding how to combine two or more machine learning models into a hybrid pipeline and integrate it with a web application.
- Individual learning issue was covered by doing technical research review.
- Multiple public datasets were explored and studied.
- Multiple malware attacks were learnt about.
- CIC-MalMem2022 was selected due to its labelled behavioural traces and memory-based features, ideal for anomaly detection use cases [7].
- Different machine learning algorithms used in malware detection were explored.
- In parallel with technical research, leadership was shown in guiding early group planning in the team.

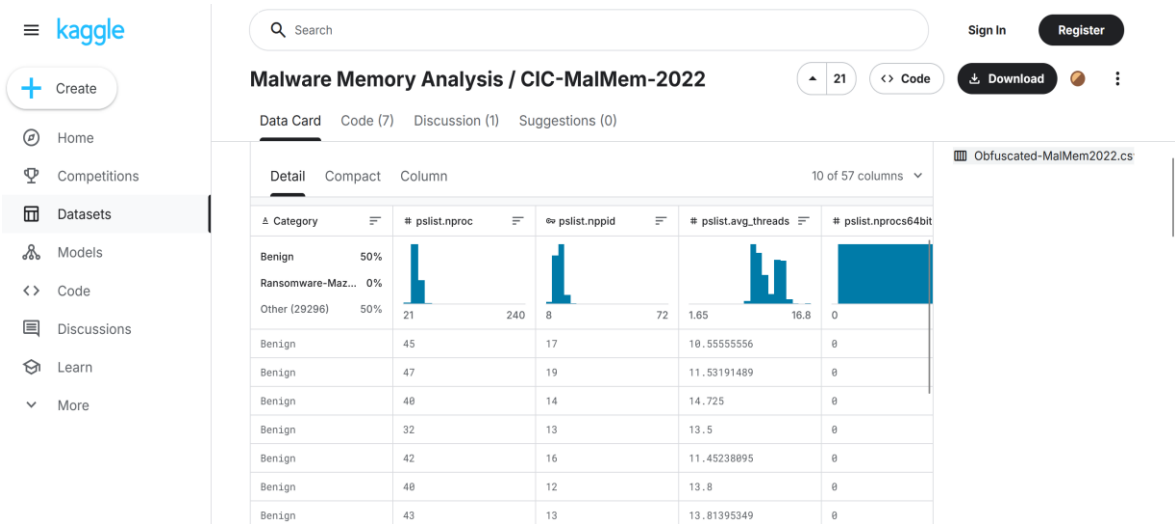


Fig 18: THE CIC-MalMem2022 dataset downloaded from Kaggle [7].

Phase 2

(Weeks 4–6) - System Design, Figma Prototype, Hybrid Model Training

- CyberShield AI (Design 1) was designed as a three-tier web system using Vue.js for the frontend, Flask for the backend, and MySQL for data storage.
- Sci-Kit, pandas and Tensorflow libraries were used for the Hybrid model pipeline [2].
- The architecture included.
 - A .csv file input and data preprocessing pipeline.
 - A 7-feature Autoencoder for anomaly detection [3].
 - Two Random Forest models for classification (binary and multi-class) [2].
 - Scan history tracking and risk score computation [5].
 - Admin-only modules for retraining and dataset updates [1].

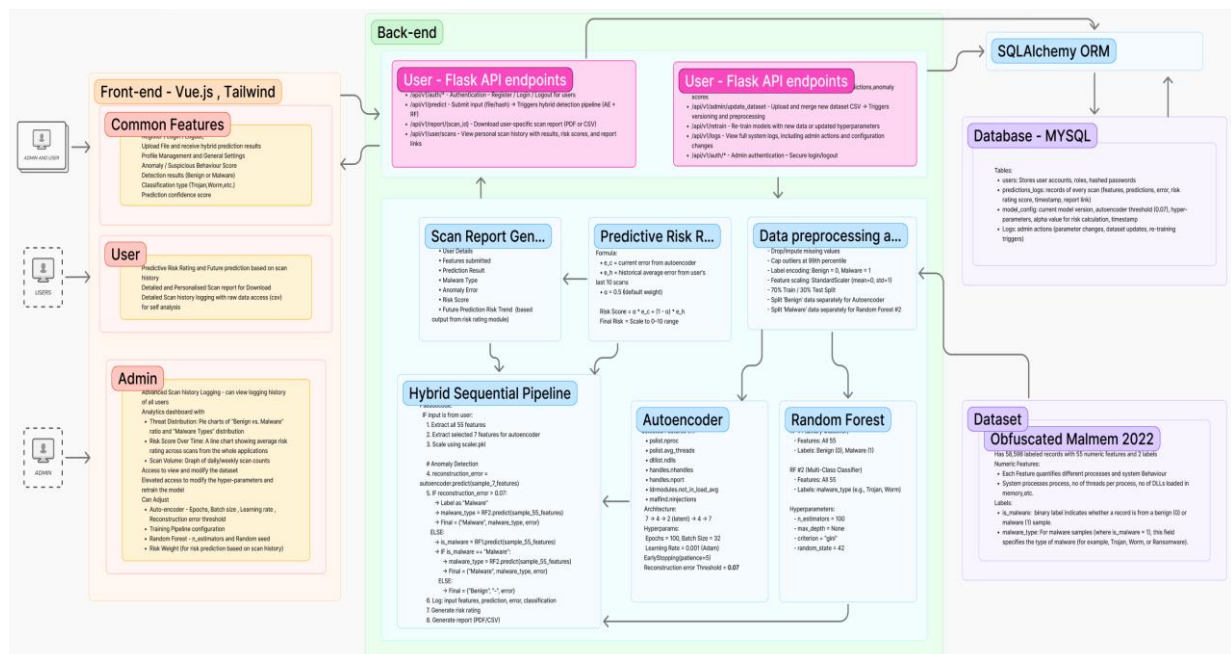


Fig 19: CyberShield AI System Architecture [1,2,3,5,6,]

- A complete Figma prototype was developed, covering all user interface pages.
- Pages for User Landing (after login), Dashboard, Scan upload, Results, Analytics and Model retrain settings.
- The design allowed the team to clearly visualise system flow and interface usability.

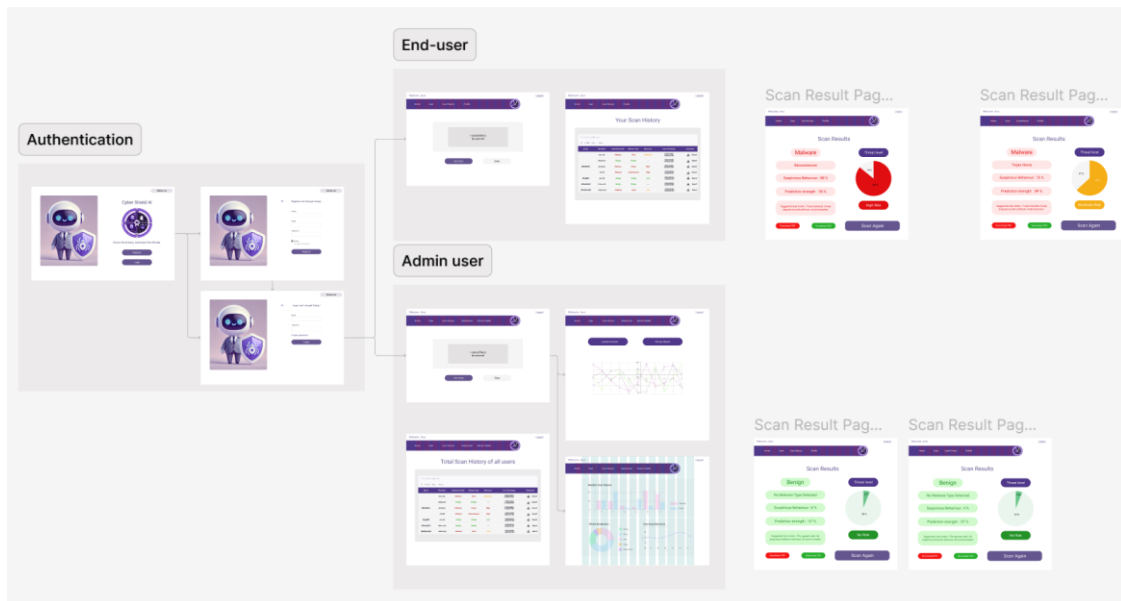


Fig 20: Design 1 Figma

- During Weeks 5–6, dataset preprocessing and initial model training were completed:
- Seven key features were selected for the Autoencoder, and all 55 features were retained for classification [3,5].
- The Autoencoder was trained on benign data to learn normal system behaviour [3,5].
- A reconstruction error threshold was dynamically set to detect anomalies [5].
- A RF binary classifier was trained for general malware detection [1,6].
- A RF multi-class classifier was trained to identify specific malware types: Trojan, Ransomware, and Spyware [1,6].
- Models were saved in .pkl format, and validation was done using confusion matrices to fine-tune prediction accuracy [2].

```
autoencoder.compile(optimizer=Adam(learning_rate=learning_rate), loss=MeanSquaredError())
early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = autoencoder.fit(
    X_train_ae, X_train_ae,
    validation_data=(X_val_ae, X_val_ae),
    epochs=epochs,
    batch_size=batch_size,
    callbacks=[early_stop],
    verbose=1
)

ae_model_path = os.path.join(MODEL_DIR, 'autoencoder_model.h5')
if os.path.exists(ae_model_path):
    os.remove(ae_model_path)
autoencoder.save(ae_model_path)
output_log.append(f" Autoencoder model trained for {len(history.history['loss'])} epochs (
```

Fig 21: Autoencoder [3,5]


```

# ---- RF1 ----
df_rf1 = pd.read_csv(RF1_DATA_PATH)
X_rf1 = df_rf1.drop(columns=['Class']).copy()
y_rf1 = df_rf1['Class'].apply(lambda x: 0 if x == 'Benign' else 1)
for col in X_rf1.columns:
    upper = X_rf1[col].quantile(0.99)
    X_rf1.loc[:, col] = np.where(X_rf1[col] > upper, upper, X_rf1[col])
scaler_rf1 = StandardScaler()
X_rf1_scaled = scaler_rf1.fit_transform(X_rf1)
with open(os.path.join(MODEL_DIR, 'scaler_rf1.pkl'), 'wb') as f:
    pickle.dump(scaler_rf1, f)
X_train_rf1, _, y_train_rf1, _ = train_test_split(X_rf1_scaled, y_rf1, test_size=0.3, random_state=42, stratify=y_rf1)
rf_binary = RandomForestClassifier(n_estimators=100, random_state=42)
rf_binary.fit(X_train_rf1, y_train_rf1)
with open(os.path.join(MODEL_DIR, 'rf_binary.pkl'), 'wb') as f:
    pickle.dump(rf_binary, f)
output_log.append(f" Random Forest (binary) trained with 100 trees (Accuracy: {rf_binary.score(X_train_rf1, y_train_rf1):.2%})")

# ---- RF2 ----
df_rf2 = pd.read_csv(RF2_DATA_PATH)
X_rf2 = df_rf2.drop(columns=['Category']).copy()
y_rf2 = df_rf2['Category']
for col in X_rf2.columns:
    upper = X_rf2[col].quantile(0.99)
    X_rf2.loc[:, col] = np.where(X_rf2[col] > upper, upper, X_rf2[col])
X_rf2_scaled = scaler_rf1.transform(X_rf2) # reuse scaler
X_train_rf2, _, y_train_rf2, _ = train_test_split(X_rf2_scaled, y_rf2, test_size=0.3, random_state=42, stratify=y_rf2)
rf_multi = RandomForestClassifier(n_estimators=100, random_state=42)
rf_multi.fit(X_train_rf2, y_train_rf2)
with open(os.path.join(MODEL_DIR, 'rf_multiclass.pkl'), 'wb') as f:
    pickle.dump(rf_multi, f)
output_log.append(f" Random Forest (multi-class) trained with 100 trees (Accuracy: {rf_multi.score(X_train_rf2, y_train_rf2):.2%})")

output_log.append(" All models trained and saved successfully.")

```

Fig 22: Random Forest Classifier [1,6].

```

Epoch 1/100
733/733 ----- 3s 1ms/step - loss: 0.7770 - val_loss: 0.4693
Epoch 2/100
733/733 ----- 1s 845us/step - loss: 0.3974 - val_loss: 0.1820
Epoch 3/100
733/733 ----- 1s 1ms/step - loss: 0.1710 - val_loss: 0.1445
Epoch 4/100
733/733 ----- 1s 851us/step - loss: 0.1446 - val_loss: 0.1363
Epoch 5/100
733/733 ----- 1s 756us/step - loss: 0.1356 - val_loss: 0.1326
Epoch 6/100
733/733 ----- 1s 1ms/step - loss: 0.1341 - val_loss: 0.1299
Epoch 7/100
733/733 ----- 1s 1ms/step - loss: 0.1309 - val_loss: 0.1274
Epoch 8/100
303/733 ----- 0s 1ms/step - loss: 0.1294

```

Fig 23: Model getting Trained

Logs

Autoencoder model trained for 47 epochs (Final val loss: 0.1173) and saved.

Random Forest (binary) trained with 100 trees (Accuracy: 100.00%) and saved.

Random Forest (multi-class) trained with 100 trees (Accuracy: 99.97%) and saved.

All models trained and saved successfully.

Fig 24: Model finished training

Phase 3

(Weeks 7–9) – Backend Development and Front-end Development

- During this phase, full web development of design 1 (CyberShield AI) was completed across both backend and frontend layers.
- Flask-based REST APIs were developed to support key functionalities:
 - File upload handling, feature extraction, and preprocessing.
 - Sequential Hybrid model using the Autoencoder and Random Forest classifiers.
 - Risk score computation using the weighted formula [2]:

$$\text{future_risk} = (\text{current_anomaly_score} * 0.6) + (\text{historical_avg} * 0.4)$$

Predictive Risk Rating Module

Based on previous scan history of users

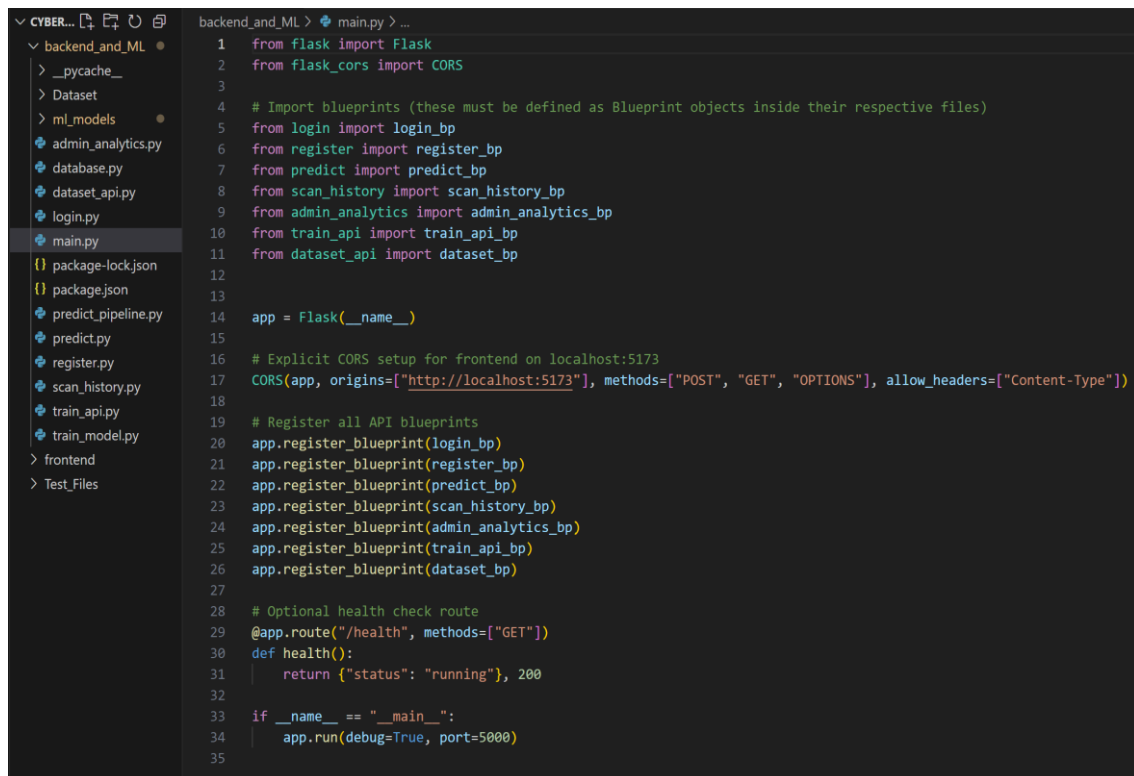
Formula:

- e_c = current error from autoencoder
- e_h = historical average error from user's last 10 scans
- $\alpha = 0.5$ (default weight)

$$\text{Risk Score} = \alpha * e_c + (1 - \alpha) * e_h$$

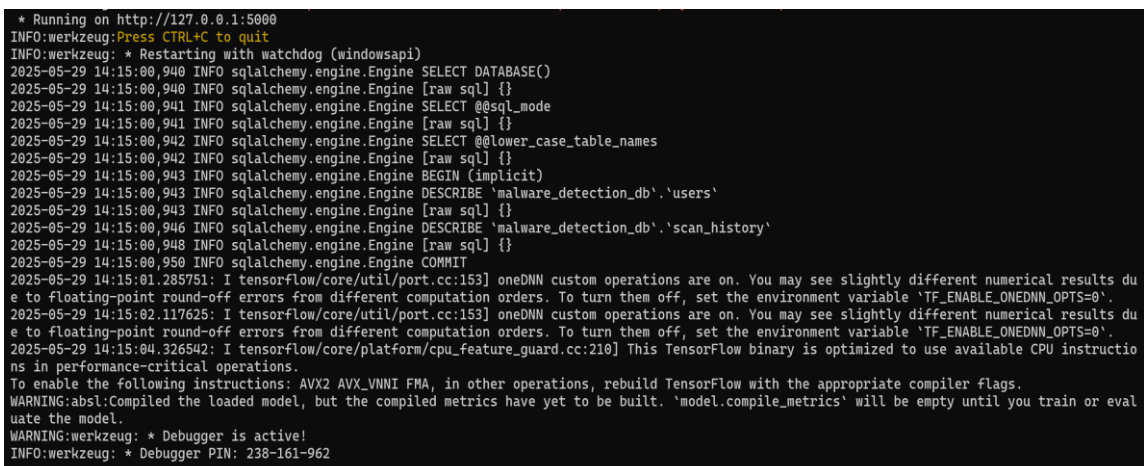
Final Risk = Scale to 0–10 range

Fig 25: Predictive Risk Rating Module [5]



```
backend_and_ML > main.py > ...
1  from flask import Flask
2  from flask_cors import CORS
3
4  # Import blueprints (these must be defined as Blueprint objects inside their respective files)
5  from login import login_bp
6  from register import register_bp
7  from predict import predict_bp
8  from scan_history import scan_history_bp
9  from admin_analytics import admin_analytics_bp
10 from train_api import train_api_bp
11 from dataset_api import dataset_bp
12
13
14 app = Flask(__name__)
15
16 # Explicit CORS setup for frontend on localhost:5173
17 CORS(app, origins=["http://localhost:5173"], methods=["POST", "GET", "OPTIONS"], allow_headers=["Content-Type"])
18
19 # Register all API blueprints
20 app.register_blueprint(login_bp)
21 app.register_blueprint(register_bp)
22 app.register_blueprint(predict_bp)
23 app.register_blueprint(scan_history_bp)
24 app.register_blueprint(admin_analytics_bp)
25 app.register_blueprint(train_api_bp)
26 app.register_blueprint(dataset_bp)
27
28 # Optional health check route
29 @app.route("/health", methods=["GET"])
30 def health():
31     return {"status": "running"}, 200
32
33 if __name__ == "__main__":
34     app.run(debug=True, port=5000)
35
```

Fig 26: Full Backend code directory from main project directory



```
* Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with watchdog (windowsapi)
2025-05-29 14:15:00,940 INFO sqlalchemy.engine.Engine SELECT DATABASE()
2025-05-29 14:15:00,940 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-05-29 14:15:00,941 INFO sqlalchemy.engine.Engine SELECT @@sql_mode
2025-05-29 14:15:00,941 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-05-29 14:15:00,942 INFO sqlalchemy.engine.Engine SELECT @@lower_case_table_names
2025-05-29 14:15:00,942 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-05-29 14:15:00,943 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-05-29 14:15:00,943 INFO sqlalchemy.engine.Engine DESCRIBE 'malware_detection_db'.users'
2025-05-29 14:15:00,943 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-05-29 14:15:00,946 INFO sqlalchemy.engine.Engine DESCRIBE 'malware_detection_db'.scan_history'
2025-05-29 14:15:00,948 INFO sqlalchemy.engine.Engine [raw sql] {}
2025-05-29 14:15:00,950 INFO sqlalchemy.engine.Engine COMMIT
2025-05-29 14:15:01,285751: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-05-29 14:15:02,117625: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2025-05-29 14:15:04,326542: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 238-161-962
```

Fig 27: Flask server successfully running

- Secure API routing using Flask Blueprints.
- Admin-only model retraining and viewing analytics dashboard.
- SQLAlchemy ORM was used to manage database interactions. Two main MySQL tables were created.
 - **users table** – storing user credentials and login details.
 - **scan_history table** – storing all scan records including prediction results, anomaly scores, and timestamps.
- Frontend development was carried out using Vue.js, based on the Figma prototype. Key UI components included:
 - ◆ A role-based login system for admin and user segregation.
 - ◆ A malware scan page with file upload and scan result display.
 - ◆ PDF report generation with all scan result details.
 - ◆ Admin dashboard with visual analytics (charts and graphs)
 - ◆ Editable dataset viewer, and input fields for retraining hyperparameters of the model [4].
- Tailwind CSS was used to ensure responsiveness and modern UI styling.
- Vue's reactive binding allowed seamless communication with backend APIs, keeping all dashboard elements dynamically updated.

The screenshot shows a code editor with a file explorer on the left and the code for `ScanFile.vue` on the right. The file explorer lists the project structure, including `backend_and_ML`, `frontend`, `node_modules`, `public`, `src`, `assets`, `components`, `router`, and `views`. Under `views`, several Vue components are listed, with `ScanFile.vue` selected. The code for `ScanFile.vue` is displayed in the main editor area, showing a template with a file upload section and a results section. The file upload section includes a label, a span for file name, and an input field. The results section includes a heading and a list of results.

```

1 <template>
2 <div class="w-screen min-h-screen bg-white flex flex-col font-['Calibri'] items-center overflow-x-hidden px-4 py-8">
3   <div class="w-full max-w-5xl">
4     <!-- Upload Section -->
5     <div class="bg-[#F3F3F3] rounded-xl p-8 flex flex-col items-center shadow-md">
6       <label for="file-upload" class="w-full max-w-xl h-40 bg-white rounded-lg border-2 border-dashed border-gray-300 f
7       <span class="text-gray-500 text-lg font-medium">
8         {{ file ? file.name : '+ Upload file to be scanned' }}
9       </span>
10      </label>
11      <input id="file-upload" type="file" accept=".csv" class="hidden" @change="onfileChange" />
12      <div class="flex justify-center gap-4 mt-6">
13        <button
14          @click="runScan"
15          :disabled="!file"
16          class="w-40 py-3 bg-[#4F378A] text-white rounded-full text-base font-semibold disabled:opacity-50"
17        >Run Scan</button>
18        <button
19          @click="clearFile"
20          class="w-40 py-3 bg-gray-300 text-black rounded-full text-base font-medium"
21        >Clear</button>
22      </div>
23    </div>
24  </div>
25  <!-- Results -->
26  <div v-if="results.length" class="mt-10">
27    <h3 class="text-xl font-semibold text-[#4F378A] mb-6 text-center">Scan Results</h3>
28
29    <div
30      v-for="(res, i) in results"
31      :key="i"
32      class="mb-8 p-6 rounded-lg shadow border border-gray-200 bg-white"
33    >
34      <div class="flex justify-between items-center mb-4">
35        <p class="text-lg font-bold text-[#4F378A]">Classification {{ i + 1 }} {{ file.fileName || 'Unnamed.csv' }}</p>
36        <button

```

Fig 28: Full Frontend code directory from main project directory

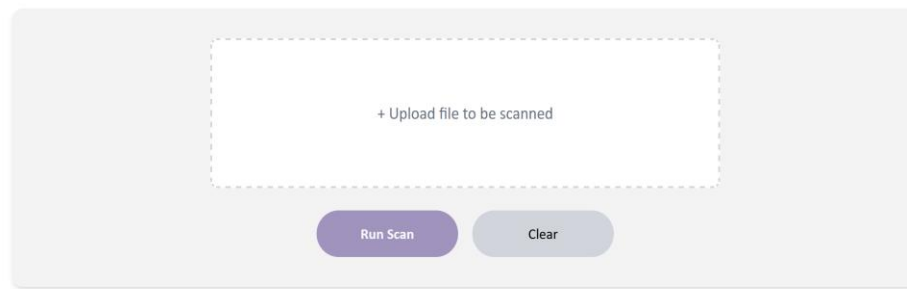


Fig 29: Front end page where user/admin can scan a file

Phase 4

(Weeks 10–12) – System Integration, Testing, Presentation, and Delivery

- Extensive testing was conducted to confirm reliability, accuracy, and proper data flow across the full system:
 - Bugs were fixed for anomaly scores.
 - Classification label mismatches were corrected and verified.
 - Admin actions were logged properly and tested through multiple scenarios.
- Scan Results after final functional Improvement are shown below:

Full functional Testing - Admin User: Arun

- Admin user uploads a.csv file and scans it.
- The system detects, predicts, classifies to find 3 different malware types on the file – Trojan, Ransomware and a Spyware.
- It also shows the anomaly score and future risk prediction rating for each detection.

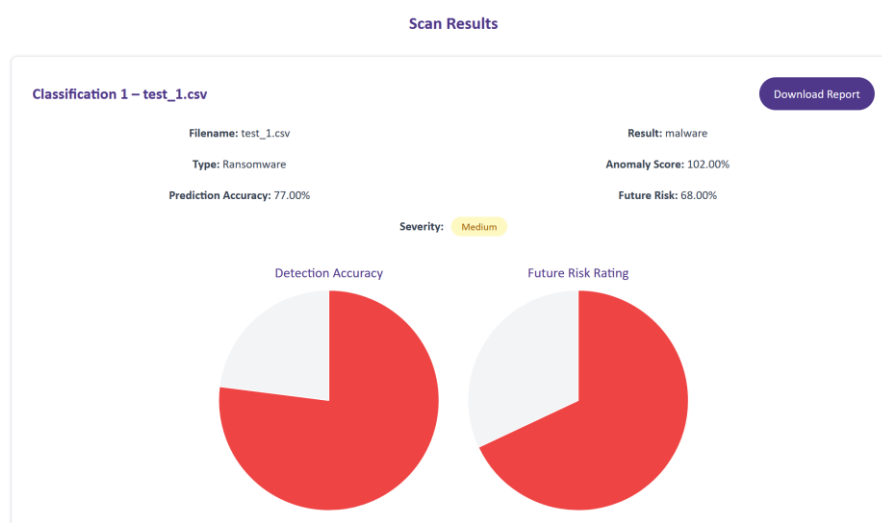


Fig 30: Ransomware malware found



Fig 31: Spyware malware found



Fig 32: Trojan malware found

- The admin then views the analytics dashboard, which shows total scans (daily, weekly, and monthly), average detection rate by malware type, and malware distribution graphs.

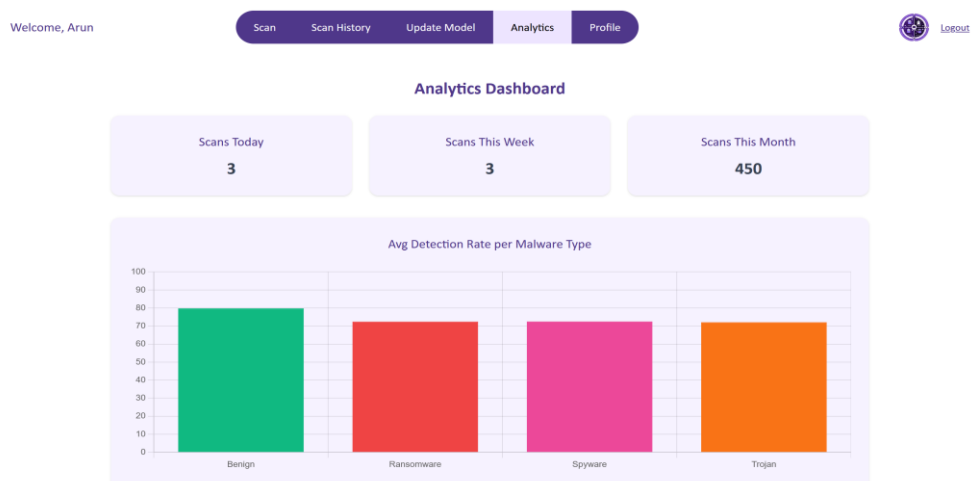


Fig 33: Analytics dashboard

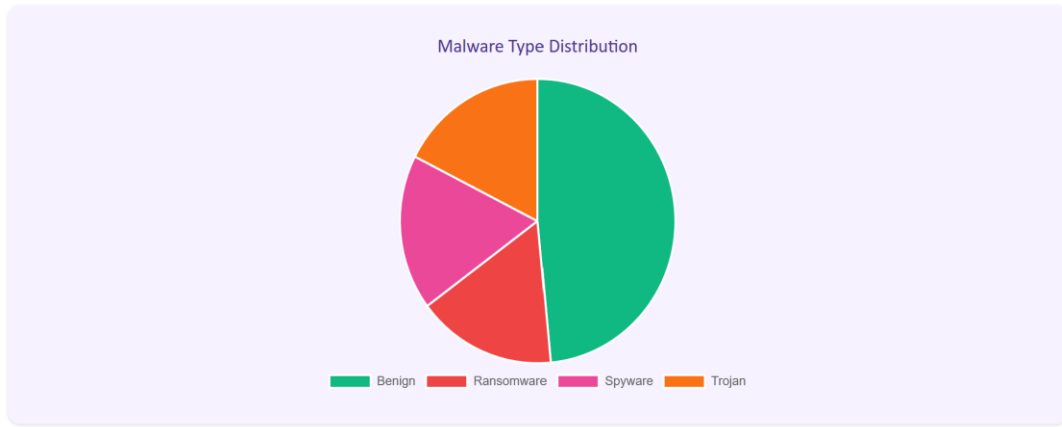


Fig 34: Analytics dashboard

- The admin then downloads the dataset, makes changes, uploads the updated one.
- The admin changes the hyperparameters (Epochs, Batch Size, Learning Rate) and retrains the model [4].

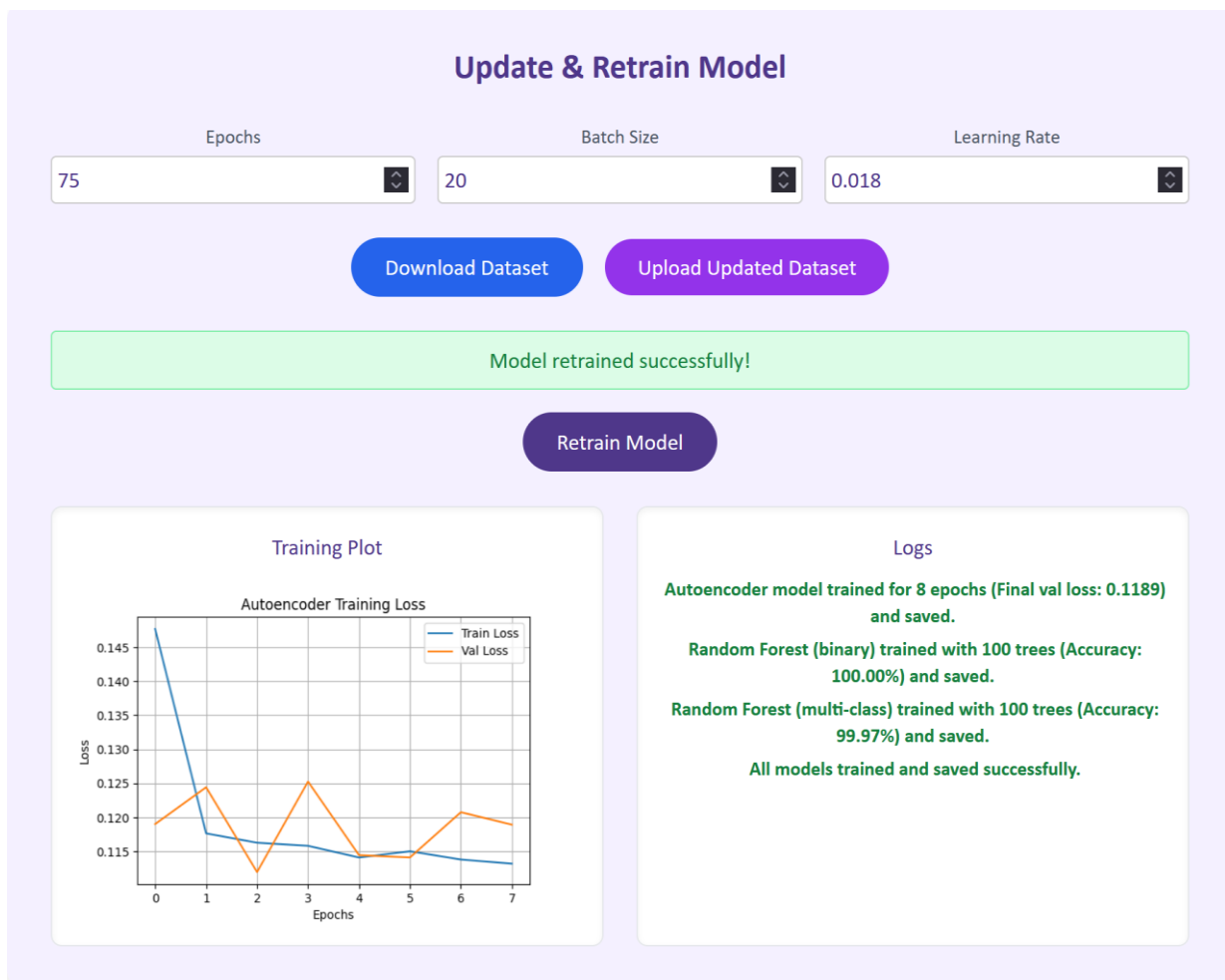


Fig 35: Admin retrains the model successfully [4]

- In Week 11 and 12, a live presentation and demonstration of CyberShield AI was conducted for the client, respectively. All key features were presented end-to-end:
 - A scan was uploaded, processed, and classified.
 - Anomaly score and risk scores were explained clearly.
 - The admin analytics dashboard was shown with real data and model retraining was showcased.

The project was delivered as a fully working end-to-end malware detection platform, fully aligned with the client requirements, and ready for further enhancement or deployment.

3.2 Individual Contributions to the Group as a whole

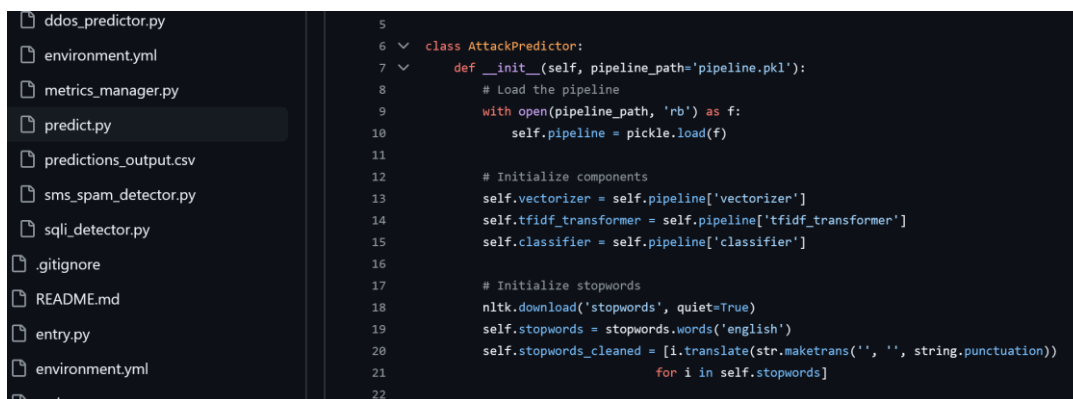
My individual technical work directly supported the success of the whole group project.

I was fully responsible for developing Design 1 – CyberShield AI. The implementation met all client requirements and was successfully tested and demonstrated.

Apart from this, I made several contributions that directly supported the group’s progress and delivery.

Evidences for things discussed below have already been provided as screenshots and discussed in detail, in the group reflection section. Linking and summarising them here in the individual reflection section, to form a complete connection.

- **Leadership and Planning:** Created the Trello taskboard and broke down weekly responsibilities for all team members. This helped keep everyone aligned and avoided delays.
- **Report Compilation:** Took full charge of cleaning, rewriting, and formatting the Innovation Concept document. Combined all five designs into one clear, consistent submission.
- **Technical Support:** Helped Tien with fixing front end component errors and frontend routing in Design 3. Supported Vatana in Design 2 by explaining model evaluation steps and helping with preprocessing.



```

5
6 class AttackPredictor:
7     def __init__(self, pipeline_path='pipeline.pkl'):
8         # Load the pipeline
9         with open(pipeline_path, 'rb') as f:
10             self.pipeline = pickle.load(f)
11
12         # Initialize components
13         self.vectorizer = self.pipeline['vectorizer']
14         self.tfidf_transformer = self.pipeline['tfidf_transformer']
15         self.classifier = self.pipeline['classifier']
16
17         # Initialize stopwords
18         nltk.download('stopwords', quiet=True)
19         self.stopwords = stopwords.words('english')
20         self.stopwords_cleaned = [i.translate(str.maketrans('', '', string.punctuation))
21                                 for i in self.stopwords]
22
  
```

Fig 36: Technical support for Vatana (Design 2 – Model)

```
502 # === User Management Endpoint ===
503 @routes.route('/api/users')
504 def get_users():
505     # Return all usernames
506     users = []
507     if os.path.exists(USER_FILE):
508         with open(USER_FILE, 'r') as f:
509             user_data = json.load(f)
510             users = [user['username'] for user in user_data]
511     return jsonify(users)
512
513 # === User Login Endpoint ===
514 @routes.route('/login', methods=['POST'])
515 def login():
516     data = request.get_json()
517     username = data.get('username')
518     password = data.get('password')
519
520     user = user_store.get(username)
521     if user and user['password'] == hash_password(password):
522         session['username'] = username
523         session['role'] = user.get('role', 'user')
524         role = user.get('role', 'user')
525     return jsonify({
```

Fig 37: Technical support for Tien (Design 3 – routing)

- **Design Reviews:** Reviewed Figma screens for all designs and worked closely with Klos to ensure consistency with the project goals.
- **Group Morale and Communication:** Pushed for team updates during slower weeks, managed live check-ins, and made sure tasks were completed before deadlines.
- All these contributions helped the team deliver accurate and efficient designs.
- I kept my work clear, updated others regularly, and stepped in when help was needed.
- This project proved that doing your part well and also helping others when they need support, can highly improve the final result of a group project.

4. Conclusion and Recommendations

4.1 Conclusion

The primary objective of this project was to build a full-stack malware detection and classification system using hybrid machine learning models. This would be supported by malware datasets and behavioural risk analysis. **CyberShield AI (Design 1)** fulfilled this goal completely.

The system delivered what the client expected:

- It applied a **two-step sequential hybrid AI pipeline**—first, detecting unusual behaviour using an Autoencoder; second, classifying malware with Random Forest models.
- It handled both **known and unknown malware types**, performing risk analysis based on anomaly scores and scan history.
- It was trained on a **real public dataset (CIC-MalMem2022)**, and fully integrated into a three-tier web application.
- It offered separate **user and admin usage**:
 - Users could scan files, view verdicts, and download scan reports.
 - Apart from regular user features, admins also had the control to retrain models, update datasets, and view scan data analytics.

All tasks, from initial research and dataset selection to model development, system integration, and testing, were completed within the project timeline of 12 weeks. The final solution is accurate, stable, and ready for future expansion.

4.2 Recommendations for Further Development

To further improve CyberShield AI for enterprise or production use, the following practical upgrades are recommended:

- **Cloud Hosting:** Deploying the system using AWS EC2 (for the app), RDS (for the database), and S3 (for secure file handling) to improve accessibility, scalability, and uptime.
- **Real-Time Detection:** Integrating live monitoring tools (e.g., using inotify on Linux or ETW on Windows) to watch system memory or disk activity instead of relying solely on uploaded files.
- **Model Versioning:** Allowing admins to tag model versions and roll back to the last working state if a retrain causes drop in accuracy.
- **Threat Intelligence Feeds:** Connecting to APIs like MISP or VirusTotal to fetch contextual threat information and alert users when scans match known indicators of compromise (IOCs).
- **Sandbox Simulation:** Integrating a local sandbox to run suspicious files in a safe virtual environment and observe runtime behaviour before final classification, especially helpful for evasive or fileless malware.
- **Parallel Model Execution:** Instead of the sequential flow, running the Autoencoder and Random Forests in parallel to reduce scan times and allow faster decision-making.
- **More File Types:** Extending scanning capabilities to handle .exe, .docx, .pdf, and .pcap files by building in safe unpackers or file extractors.
- **Secure Role Management (RBAC):** Adding user role-based access control with full logging to track admin actions, scan usage, and dataset changes—critical for audit compliance.
- **Larger Dataset Support:** Using expanded malware datasets (like EMBER or MalMem2023) to cover more malware families, improving both detection rate and generalisation.
- **Automatic Dataset Updates:** Building an update engine that allows admins to periodically pull new training data automatically, validate it, and retrain the model without manual steps.
- **Encrypted Report Storage:** Securing stored prediction results and reports using AES-256 encryption, especially for sensitive scans from enterprise users.

Together, these additions would turn CyberShield AI into a scalable, cloud-ready, and highly secure system, ready for real-world deployment in schools, startups, or security teams.

5. References

- [1] M. Alazab, A. Awajan, A. Abdallah, H. Alqahtani, and A. Alzahrani, "Intelligent forensics: AI-based threat hunting and digital forensics in cyberspace," *Future Generation Computer Systems*, vol. 128, pp. 84–104, 2022. doi: 10.1016/j.future.2021.11.008.
- [2] N. Idrees, M. Asim, M. Aslam, A. Deebak, and H. W. Kim, "Secure and efficient malware detection using hybrid convolutional neural networks," *IEEE Access*, vol. 10, pp. 15717–15729, 2022. doi: 10.1109/ACCESS.2022.3148972.
- [3] C. M. Chen, W. C. Peng, and L. F. Chien, "Malware detection using deep learning with memory forensics," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 391–403, 2022. doi: 10.1109/TIFS.2021.3133426.
- [4] S. R. Moosavi, A. Jolfaei, S. Shamshirband, and M. K. Khan, "Botnet detection by monitoring data stream using deep learning," *IEEE Access*, vol. 8, pp. 212951–212961, 2020. doi: 10.1109/ACCESS.2020.3040459.
- [5] N. Milosevic, A. Dehghantanha, and K. R. Choo, "Machine learning aided Android malware classification," *Computers & Electrical Engineering*, vol. 61, pp. 266–274, 2017. doi: 10.1016/j.compeleceng.2017.03.006.
- [6] L. Vinayakumar, K. P. Soman, and S. Poornachandran, "Applying deep learning approaches for network traffic prediction and classification," *IEEE Access*, vol. 6, pp. 51138–51152, 2018. doi: 10.1109/ACCESS.2018.2868595.
- [7] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, p. 102526, 2020. doi: 10.1016/j.jnca.2019.102526.
- [8] CIC, "CIC-MalMem2022 Dataset," Canadian Institute for Cybersecurity, 2022. [Online]. Available: <https://www.unb.ca/cic/datasets/malmem2022.html>