

Milestone Progress Evaluation: Milestone 2

1. Project Title: Crescent Social Guardian

2. Team Names and Emails:

- Adrianna Sauro, asauro2022@my.fit.edu
- Bhavik Patel, patelb2024@my.fit.edu
- Michael Wahoski, mwahoski2020@my.fit.edu
- Nicholas Blaisdell, nblaisdell2024@my.fit.edu

3. Faculty Advisor and Client: Dr. King, michaelking@fit.edu

4. Progress of current Milestone:

Task	Completion %	Adrianna Sauro	Bhavik Patel	Nicholas Blaisdell	Michael Wahoski	To do
1. Small Demos	100%	25%	25%	25%	25%	None
2. Implement, test & demo model pipeline	60%	50%	50%	0%	0%	Testing & Demo of full pipeline
3. Implement, test & demo extension	80%	0%	0%	0%	100%	Adding multiple structures to the test harness to test adapter standardization
4. Implement, test & demo adapter	80%	0%	0%	90%	10%	Optimization
5. Implement, test & demo the classification model	5%	50%	50%	0%	0%	Implement, test & demo

5.

- Task 1:

This task involved developing demonstrations for the milestone presentation to showcase the initial functionality of the integrated pipeline. The demonstration showed the extension receiving messages from the test harness in real time, adds the location where the chat is pulled from (test harness, instagram, etc), converts the pulled messages to a json structure, and verifying there is no duplicate message output before forwarding the output to the adapter. The adapter then receives this output from the extension, verifies the platform it's coming from and chooses a normalization method based on that whichever method is chosen normalizes different platform builds into a standard one. Next, the adapter generated a JSON file with all the data in the standardized format.

Another small demo shows the models ability to take in the JSON file to run EmotionBERT and DistilBERT on the message window provided by the adapter. The primary objective of the demonstration was to establish that a functional end-to-end pipeline is in place. Specifically, it demonstrated that the extension can capture real-time chat messages and forward them to the adapter, that the adapter can process those messages and produce output in a model-compatible format, and that the model can ingest the resulting JSON file and run inference on logs collected through the pipeline. In the future, the adapter will be able to send a JSON object and the model interface rather than an entire file to minimize memory usage.

- Task 2:

Task 2 involves implementing, training, testing, and preparing a demonstration of the model pipeline for CRESCENT Social Guardian. Building on the previous milestone, our work this semester has included implementing EmotionBERT, which extracts the emotions or feelings that are being conveyed in a specific message window, and DistilBERT, which reads messages and converts the words into numerical vectors that capture their contextual meaning. We do not need to train EmotionBERT as it is pre-trained to complete the task. We do need to fine-tune DistilBERT to properly analyze the tokenized messages for our task, and our custom window-level classifier head to appropriately classify the merged output of DistilBERT and EmotionBERT. The demonstration described in Task 1 shows how the current implementation processes a single chat window, while support for processing multiple windows across a conversation remains under development.

During this milestone, our faculty advisor emphasized that analyzing individual windows is not sufficient for the conversation-level contextual analysis our project requires. We also need to examine how patterns develop across successive windows

over time. Consequently, a substantial portion of this milestone focused on researching different temporal analysis models such as, Long Short-Term Memory (LSTM) networks and Bidirectional Gated Recurrent Units (BiGRUs), as potential additions to our pipeline. These recurrent approaches support sequence modeling, which makes them relevant to examining relationships across ordered conversation windows. Our research has focused on comparing these options and determining how a temporal component could use the BERT models' outputs to analyze the progression of a conversation, rather than treating each window independently. This temporal component remains in the research stage.

A major obstacle has been coordinating our team's schedule with our faculty advisor's availability. Receiving feedback early enough to research architectural changes, implement them, and test them before the milestone deadline has been challenging. Although the BERT components are implemented and training is nearing completion, incorporating the temporal analysis component remains an important next step before we can demonstrate the full conversation-level pipeline.

- Task 3:

The browser extension was developed to extract chat messages from the test harness and transmit them to the adapter for processing. The extension uses JavaScript and a DOM reader to identify and collect message information, including the message ID, sender, text, and timestamp. A MutationObserver was implemented to monitor changes within the chat interface, allowing the extension to detect and process newly added messages. The extension also tracks previously processed messages to prevent duplicate transmissions. The extension was successfully connected to the adapter using a sender script that transmits the collected information in a consistent JSON structure. The manifest file was configured to load the necessary scripts and establish the extension's functionality within the browser.

Several challenges occurred during development, including script loading errors, inconsistent naming conventions, and difficulties establishing communication between the extension and adapter. Another challenge involved detecting when users switched between conversations, requiring additional changes to ensure the extension could recognize and process messages from different chats. Currently, the extension successfully extracts and transmits messages from the test harness to the adapter.

Future development will focus on supporting additional chat platforms, including Instagram and Discord, which will require platform-specific methods for identifying and extracting messages.

- Task 4:

The adapter was developed using Python, FastAPI, Uvicorn, and Pydantic to receive, validate, normalize, and sanitize chat data transmitted from the browser extension. The POST endpoint, server address, port, and route were successfully configured to establish communication between the extension and adapter. Both components use consistent JSON nomenclature to ensure compatibility when transmitting and receiving message information. FastAPI and Uvicorn were successfully integrated to handle incoming requests, while Pydantic models were implemented to validate the message and conversation payloads. Middleware was configured to support communication with the browser extension, and error handling was implemented to identify issues involving incoming data and connection requests.

Several difficulties occurred during development, particularly with inconsistent naming conventions between the extension and adapter. Initial testing also presented challenges when running both components simultaneously and ensuring updates were properly reflected after changes to the code. The adapter was successfully tested using JSON data transmitted from the test harness. Normalization and sanitization functionality remains in development because the current test harness already provides information in the expected structure. As a result, most message data remains unchanged during processing, with timestamp formatting being one of the primary modifications. The adapter is being structured to support different platform formats through separate processing methods. Future development will focus on expanding these methods to accommodate additional test harness formats and real-world messaging platforms.

- Task 5

Task 5 involves implementing, testing, and preparing a demonstration of the classification model for CRESCENT Social Guardian. This component will use the outputs of the transformer models and the temporal analysis to generate classifications that users can more easily interpret. Progress on this task has been limited by its dependency on the implementation work in Task 2. With EmotionBERT and DistilBERT now selected as the transformer models for the project, we have identified the need for a window-level classification head that can incorporate outputs from both models to produce a classification. This task remains in the early implementation phase, with current efforts focused on developing the classification head to support this approach.

We have also decided on our model pipeline approach, which begins with taking in a window of messages from the adaptor and feeding that data into DistilBERT and EmotionBERT. Once we receive those results, we will then merge them and feed the merged results into the window-level classification model which will return a vector or a

label that represents the predatory behavior of that specific window. Once a certain number of windows have been received, we will feed them sequentially into a temporal analysis model to show how potential predatory patterns develop over the course of the conversation. The results of this model will be fed into a conversation-level classification model to detect predatory content within the conversation. Each time a specified number of new conversation windows is stored in the database, the temporal analysis model will be called. Each call will process all windows stored for that conversation in chronological order, not just the newly added windows, to show how potential predatory patterns develop over time.

6. Discussion of contribution of each team member to the current Milestone: (at least a paragraph)

- Adrianna Sauro: I coordinated meeting schedules, established goals, and set team deadlines to maintain consistent progress. For task 1 I collaborated with Bhavik to develop the demonstration script. We also determined that EmotionBERT should analyze each author's messages separately to provide clearer insight into each participant's emotional expressions within the conversation. For task 2 I worked with Bhavik to establish a clearly defined model pipeline, documenting the purpose of each component to support a successful transition to classification-head development. Following discussions with our faculty advisor, we identified the need to incorporate temporal analysis alongside the transformer outputs to better capture contextual information across the conversation. We researched this approach and incorporated it into our pipeline design. Implementation of the temporal analysis component is ongoing. For the task Bhavik and I have begun researching the requirements for the classification component based on our updated model pipeline. This work focuses on determining how the classification head will incorporate outputs from both BERT models and the temporal analysis component.
- Bhavik Patel: I worked with Adrianna on developing the EmotionBERT pipeline that takes in the raw message data from the adaptor and then extracts the emotional context of the data. I also built an EmotionBERT script to extract the emotional context of each window within each conversation and store them along with the tokenized version of the messages themselves.
- Nicholas Blaisdell: I built the adapter that connects the extension with the rest of the backend. This demo of the adapter is used to connect the extension only, but some skeleton code was tested for future model interface connectivity. I built the normalization and sanitization functions for the adapter to clean code into the JSON format the model interface expects. I built the message payload and conversation payload to use in the extension request handler, which properly receives the JSON from the extension. I worked on the demo of the adapter in

connection with the extension. I added print statements to show the adapter connecting and receiving sent files from the extension. The adapter shows the logs that were sent, shows the startup and end of the adapter activity. There are print statements for prenormalized data, postnormalized data, presanitized, and postsanitized.

- Michael Wahoski: I finished building the alpha version of the test harness to verify pipeline connection. It is what we will use to push synthetic chat messages from extension to adapter. Its basically a chat message site that we can pull chats from and test the pipeline from extension to adapter and later on, the model. I built the extension which is a google chrome extension that reads DOM changes using mutationObserver, packages the chat into a json file, and sends it to the adapter through a rest API. I assisted with the adapter connection and helped develop required structure, debugging info, and intended input/output requirements.

7. Plan for the next Milestone

Task	Completion	Adrianna Sauro	Bhavik Patel	Michael Wahoski	Nicholas Blaisdell
1. Small Demos	100%	25%	25%	25%	25%
2. Implement, test & demo model pipeline	100%	50%	50%	0%	0%
3. Implement, test & demo the classification model	10%	50%	50%	0%	0%
4. Implement, test & demo extension	100%	0%	0%	100%	0%
5. Implement, test & demo adapter	100%	0%	5%	10%	85%
6.. Implement and Test Summarization Engine.	25%	15%	15%	0%	70%

8. Discussion of each planned task for the next Milestone

- Task 1:

For the next milestone, we will prepare brief demonstrations, similar to those presented at previous milestones, to highlight our progress and showcase the project's current functionality. Our goal for the next milestone would be to demonstrate the adapter will demonstrate accepting the extension as it has changed, accepting a diversity of inputs from the updated test harness. The extension will be demoed at the same time as the adapter. The adapter will show proper print statements for processes as they occur and will show the model interface connection and data being sent.

The explainability layer demo will show consistent print statements as the pipeline connects together, showing the user-friendly explanation from the initial tensor data. In addition we aim to demonstrate a full window analysis which includes the output from EmotionBERT and DistilBERT with a custom classifier head. We also plan to demonstrate the output of the BERT models being sent to BiGRU for temporal analysis, and what output from temporal analysis we are getting for overall conversation analysis.

- Task 2:

For the next milestone, we will complete training our model pipeline with the window implementation, and validate with the Pan12 testing dataset. We also need to select and start implementing our temporal analysis model. To support time-series training, we must aggregate the conversation windows into an appropriate format. We also need to build our model interface which will be used to regulate our model pipeline and accept incoming data from the adapter in a JSON format. From there the messages will be extracted and run through EmotionBERT, DistilBERT, and our window-level classifier head to determine whether that specific window contains any type of predatory behavior. The interface will also send the window-level classifier's results back to the database, linking each result to its corresponding conversation window. Each time a specified number of new windows has been collected, all stored windows and their associated results will be passed to the temporal analysis model in chronological order.

- Task 3:

For the next milestone, we will have a window-level classification model that will be trained to take in the results of the EmotionBERT model along with the results of the DistilBERT model and assign an appropriate score to each conversation window. These scores will be stored in our database to later be fed into our temporal analysis model to determine the overall trend of predatory behavior within the conversation. Every time our temporal analysis model is called, we will then feed that result into our conversation-level classifier head to assign a label that states whether the current state of the conversation is predatory.

- Task 4:

For this milestone, we will expand the test harness allowing different format structures to flow through the extension. There will be more formats added to the test harness that have similar structures to known platforms such as instagram. This will assist us in verifying sanitization/ standardization is happening with multiple types of structures based on labeling and platform. Optimization of both the extension and adapter will be done as updates are made, ensuring there are no lapses in connection and that data remains intact.

- Task 5:

For this milestone, the adapter will be adjusted to work with the model interface. This connection will allow incoming message data to be sent to the model from the extension. The adapter will test new inputs from the test harness and ensure proper sanitization and normalization occurs. The adapter will receive a variety of inputs from the extension and will be tested. Optimization and print statements will be used in demos similar to the Milestone 2 demos, but will show more from sanitization and normalization, as well as showing the other test harness formats changing being accepted.

- Task 6:

The summarization engine will be merged properly, linking originally separated demos and ensuring the pipeline connects without issue. Transformers Interpret will link with the BART formatter and BART to produce the summarization information for the user. Demos for this will be tested, showing print statements for each phase that passes throughout the pipeline. If possible, the model will be linked, otherwise will use similar data to the first demo.

9. Dates of meetings with Client during the current milestone:

See Faculty Advisor meetings below

10. Client feedback on the current milestone:

See Faculty Advisor Feedback below

11. Dates of meeting with Faculty Advisor during the current milestone:

- 10/7/26 12-1pm
- 10/23/26 12-1pm

12. Faculty Advisor feedback on each task for the current Milestone

- Task 1: N/A

- Task 2: Our model choice of using DistilBERT and EmotionBERT was valid, however, the addition of temporal analysis is necessary. The BERT models should be used for window level analysis, but conversation level analysis should use a temporal analysis model. The temporal model should take in a certain number of sliding windows, which update incrementally upon receiving a specified number of new messages, at a time and return a vector or score of the current state of the conversation. We were also told to analyze model output for such unbalanced data.
- Task 3: N/A
- Task 4: N/A
- Task 5: We have been advised to add a window level classifier and a conversation level classifier. Window-level output will be used to determine any sudden spikes in predatory behavior, and should a spike be detected it will immediately be reported. For conversation-level output we will be using a classification model to determine whether a conversation is predatory or non-predatory based on the results of the other models in our pipeline. We were told to research more into Jev AI, Naive-Based classifier's, decision-tree classifiers, and other classification models to determine which one would be the best for our specific task.

Faculty Advisor Signature: _____ Date: _____

13. Evaluation by Faculty Advisor

- Faculty Advisor: detach and return this page to Dr. Chan (HC 209) or email the scores to pkc@cs.fit.edu
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Adrianna Sauro	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Bhavik Patel	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Michael Wahoski	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Nicholas Blaisdell	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

Faculty Advisor Signature: _____ Date: _____