جامعة الأخوين

ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵓⵅⴰⵡⵉⵏ

AL AKHAWAYN UNIVERSITY

SCHOOL OF SCIENCE & ENGINEERING SPRING 2026

AI-Based Story Point Estimation for Agile Software Development

CSC 5382 Artificial Intelligence for digital transformation

REALIZED BY:
AMIRA HADIMI

SUPERVISED BY: DR. ASMAE MOURHIR

1. Introduction

This milestone focuses on building a working Proof of Concept (PoC) for automated story point estimation using a baseline large language model. The objective is to demonstrate successful integration of a pre-trained LLM with task-specific adapters, deploy an interactive web application using Streamlit, and evaluate the system in an end-to-end testing scenario.

Story point estimation is a core task in Agile software development, where teams assign effort values to user stories. This project leverages a large language model (Llama-based architecture) adapted for regression to predict story points directly from issue titles. The milestone demonstrates that the model can be integrated, deployed, and evaluated on real-world data.

The deliverables of this milestone include:

- Integration of the Llama3SP baseline model
- Development of an interactive Streamlit application
- End-to-end evaluation using Mean Absolute Error (MAE)
- Reproducible results stored in the GitHub repository

2. Model Integration

1. Baseline model

The baseline used is **Llama3SP**, which is based on:

- Base model: Llama-3.2-1B
- Adaptation method: LoRA (Low-Rank Adaptation) via PEFT
- Task type: Regression (predicting continuous story point values)

The base Llama model is augmented with project-specific LoRA adapters that have been fine-tuned for story point estimation. A regression head (with num_labels = 1) is used to output a single continuous prediction representing the estimated story points.

The system uses a **title-focused configuration**, where:

- The input text is truncated to a maximum length of 20 tokens.
- Tokenization is handled using HuggingFace's AutoTokenizer.
- Padding tokens are explicitly configured to avoid batch inference issues.

2. System architecture

The prediction pipeline follows the architecture below:

Issue Title
→ Tokenizer
→ Llama-3.2-1B Base Model

→ LoRA Adapter (Project-Specific)
→ Regression Head
→ Predicted Story Points

This architecture allows efficient adaptation of a large language model without fine-tuning the full base model, reducing computational requirements while preserving performance.

3. Model Integration Process

The integration of the Llama3SP model required several configuration steps:

1. **Loading the base model**
   The Llama-3.2-1B model was loaded using HuggingFace Transformers with a regression configuration:
   - num_labels = 1
   - problem_type = "regression"
2. **Loading project-specific LoRA adapters**
   Each project-specific adapter was loaded using the PEFT library and attached to the base model dynamically during evaluation.
3. **Tokenizer configuration**
   - Explicit padding token configuration was required.
   - pad_token was set to eos_token to allow batch inference.
   - Tokenization was configured with max_length = 20 to match the baseline artifacts.
4. **Device configuration**
   The model was configured for CPU inference using:
   - torch.float32
   - device_map = None
   - limited thread parallelism for Windows stability.
5. **Batch evaluation support**
   The integration supports batch processing for evaluation across multiple projects, enabling automated computation of MAE.

These steps ensured that the pre-trained base model and LoRA adapters were properly integrated into a functional regression system.

The model loading and inference testing are implemented in:

- test_llama3sp.py

3. Streamlit

To demonstrate practical usability, an interactive web application was developed using Streamlit. The goal of the application is to allow a user to input an issue title and receive a predicted story point estimate.

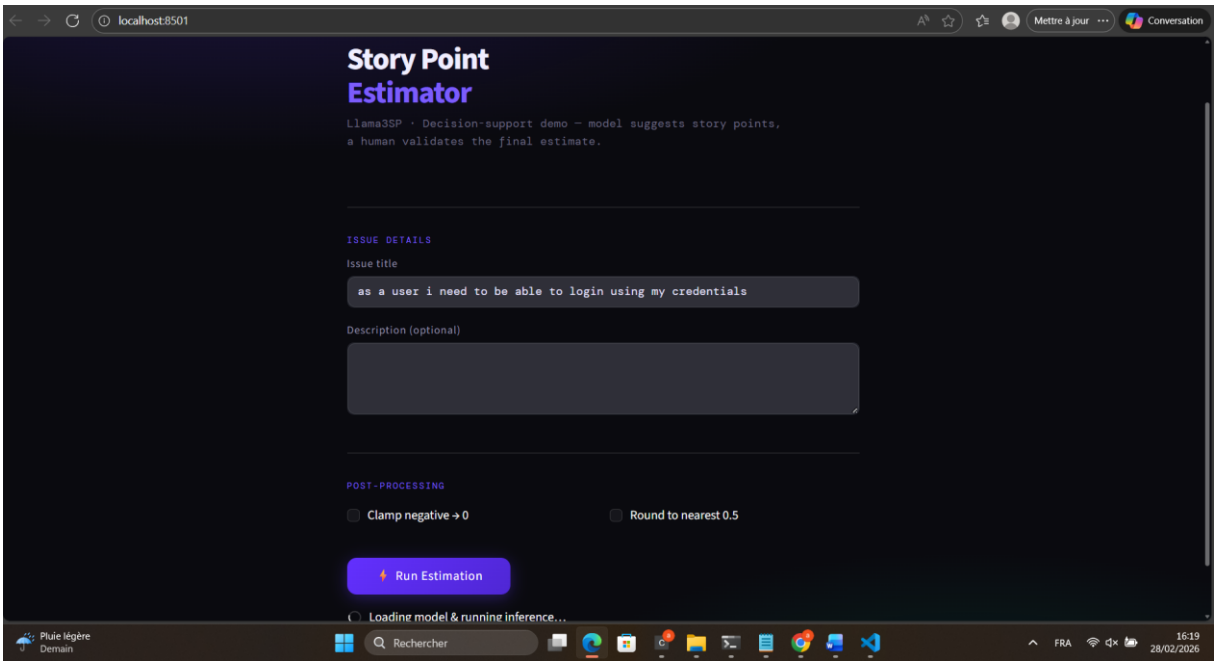
1. Application Functionality

The Streamlit application performs the following steps:

1. Accepts user input (issue title)
2. Tokenizes the input text
3. Loads the appropriate Llama3SP model and adapter
4. Generates a story point prediction
5. Applies optional post-processing:
    - Clamping negative predictions to zero
    - Rounding to the nearest 0.5
6. Displays the predicted result

The implementation is available in:

- app/streamlit_app.py



4. End to End evaluation

For a complete end-to-end testing demonstration, including live inference and evaluation walkthrough, please refer to the recorded presentation video. The video illustrates the full

workflow from user input in the Streamlit application to prediction generation and MAE computation.

1. Dataset

The model was evaluated on 16 open-source software projects from the Llama3SP dataset. For each project:

- The test split was selected using the split_mark column.
- Up to 100 test samples were used per project.
- The issue title was used as model input.

Table : example of dataset from DURACLLOUD-10 project

	TITLE	DESCRIPTION	STORYPOINT	SPLIT_MARK
DURACLOUD-10	Allow for simple updating of project version numbers	It should be easy/simple to update the version number of all parts of the baseline for each release.	3	train
DURACLOUD-19	Bulk load: Verify successful DuraCloud ingest of 10TB of BHL content		16	train
DURACLOUD-21	JPEG2K: Image conversion service	Service wrapper and service to perform conversion of images from TIFF to JPEG2K.	8	train

2. Evaluation Metric

The performance metric used is **Mean Absolute Error (MAE)**:

MAE = (1/n) Σ |y_true − y_pred|

MAE measures the average absolute difference between predicted and true story points. Lower MAE indicates better estimation accuracy.

3. Results

The evaluation was conducted across all 16 projects. The results are summarized in:

- results/mae_per_project.csv
- results/summary.json

Example results:

project	test_size	MAE
appceleratorstudio	100	1.8537879490852356
aptanastudio	100	3.8436859107017516

bamboo	100	1.0968187268823386
clover	77	4.075433501949558
datamanagement	100	6.185642478764057
duracloud	100	1.0613428312540054
jirasoftware	71	2.0503330902314523
mesos	100	1.3783015301823616
moodle	100	11.302529972791671
mule	100	2.4297026617825033
mulestudio	100	3.4888235354423522
springxd	100	1.672186747789383
talenddataquality	100	3.604175963997841
talendesb	100	1.1015135708451271
titanium	100	2.832799241542816
usergrid	97	1.5066721900222229

The results show that the model performs well on several projects (MAE ≈ 1–2), but performance varies across domains. Some projects such as Moodle exhibit higher error, potentially due to domain differences and title truncation constraints.

Conclusion

The results show that the model performs well on several projects (MAE ≈ 1–2), but performance varies across domains. Some projects such as Moodle exhibit higher error, potentially due to domain differences and title truncation constraints.