EXPENSE REPORT FEEDBACK

A MINI PROJECT REPORT

submitted by

NIRMALISWARRAN S R (715521104028)

THANISHA SHIMNAZ N (715521104051)

VISHNU KEERTHAN S (715522104059)

AARTHI S (715522244001)

Department of Computer Science and Engineering

PSG Institute of Technology and Applied Research

Coimbatore 641 062

in collaboration with

SAP Labs, Bengaluru

MAY 2024

BONAFIDE CERTIFICATE

Certified that the mini-project titled **"Expense Feedback"** is the bonafide work of **"Nirmaliswarran SR (715521104028), Thanisha Shimnaz (715521104051), Vishnu Keethan S (715522104059), Aarthi S (715522244001)"** which is carried out under our mentoring.

SIGNATURE

INDUSTRY MENTOR
Ms.Prerna Jain
Spend Engineering,
SAP Labs
Bengaluru,
Karnataka, India

SIGNATURE

ACADEMIC MENTOR
Dr. Priya Ponnuswamy
Assistant Professor
Computer Science and Engineering
PSG Institute of Technology and
Applied Research,
Coimbatore – 641 062

SIGNATURE
Srivatsan Santhanam
VICE PRESIDENT
Spend Engineering,
Head of Concur R&D,
SAP Labs
Bengaluru,
Karnataka, India

SIGNATURE
Dr. R. Manimegalai
HEAD OF THE DEPARTMENT
Professor
Computer Science and Engineering
PSG Institute of Technology and
Applied Research,
Coimbatore – 641 062

Submitted for evaluation on _____

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our beloved Principal, **Dr. N SARAVANAKUMAR, B.E.(Hons), M.Tech, Ph.D,** for his overwhelming support and encouragement on this project.

We extend our indebted to **Mr. SRIVATSAN SANTHANAM,** Vice President, Spend Engineering, Head of Concur R&D, SAP Labs India for the opportunity and support which was instrumental in the completion of this project.

We are greatly indebted to **Dr. R. MANIMEGALAI, M.E, Ph.D** Head of the Department, Computer Science and Engineering for her guidance and continuous support which was instrumental in the completion of this project.

We extend our thanks to our industry mentor, **Ms. PRERNA JAIN** SAP Labs, Bengaluru, India and academic mentor **Dr. PRIYA PONNUSWAMY, M.E, Ph.D** Assistant Professor, Computer Science and Engineering for his technical support and constant supervision without which we could not have completed this project study.

Nirmaliswarran S R
Thanisha Shimnaz N
Vishnu Keerthan S
Aarthi S

TABLE OF CONTENTS

ABSTRACT

The goal of the Expense Feedback project is to make it easier to assess expense documents and make sure they follow SAP's (Systems, Applications, and Products in Data Processing) standards. This project's goal is to accurately evaluate cost reports by categorizing them into pre-established groups including lodging, meals, travel, and other expenses and deciding if they qualify for reimbursement or not.

The project makes use of a small language model (SLM) that was created especially for the purpose of classifying feedback in relation to SAP document norms. The expenditure Feedback SLM is specifically designed to manage the unique complexities and requirements of expenditure document review inside SAP rules, in contrast to other AI generators that have a wider range of applications.

In order to meet the specific needs of evaluating SAP expense documents, a specialist tool was needed, which is why the Expense Feedback SLM was introduced. Utilizing a specialized language model, businesses can reduce the possibility of mistakes and inaccuracies that are frequently linked to broader AI generators, improving the effectiveness and precision of expense management procedures.

Keywords: Small language models,expense feedback.

LIST OF ABBREVIATIONS

SLM	Small Language Model
JS	Javascript
API	Application Programing Interface
UI	User Interface
HTML	Hyper Text Transfer Protocol
CSS	Cascading Style Sheet

CHAPTER 1

INTRODUCTION

1.1 PROJECT MOTIVATION

The “Expense feedback " gives feedback about the expense report submitted by the employees.This uses small language models which specifically focus on generating feedback for expenses which makes that more efficient than large language models .In recent years the slm are mostly used and which are efficient and performs better than llm .

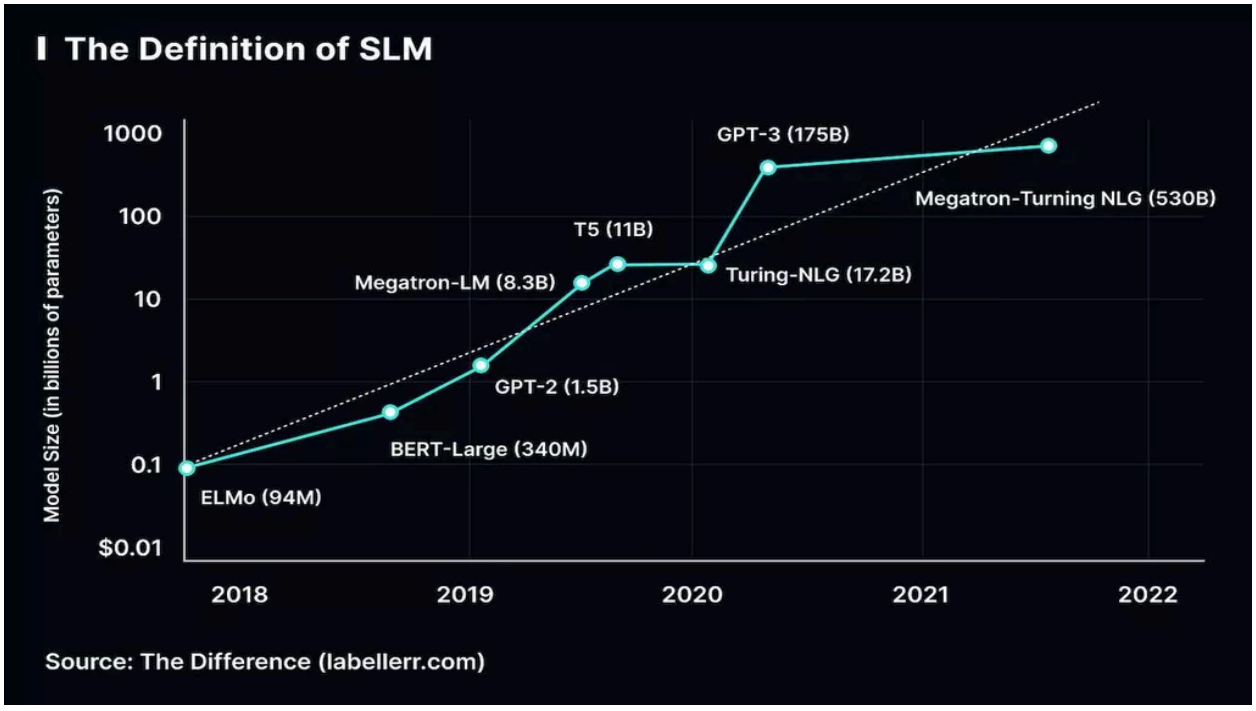

Fig 1.1.1 Increased size of models.(https://www.metadialog.com)

From Figure 1.1.1, it can be observed that at the beginning of 2018, the number of parameters was around 94 million, gradually increasing and reaching approximately 530 billion parameters by 2022. It is impractical to implement

models of such immense scale in specific fields due to their computational demands and resource requirements. Therefore, small language models offer a viable alternative, as they are fast, efficient, and easily customizable.

1.2 PROBLEM STATEMENT

Many expenditure reports are flagged throughout the current submission process due to concerns such as insufficient information, erroneous data, and the inclusion of non-reimbursable charges. These concerns cause approval process delays, an increase in administrative workload, and employee unhappiness. Without a consistent feedback process, employees struggle to comprehend why their submissions are rejected and how to improve them, resulting in recurring errors and inefficiencies.

1.3 OBJECTIVES

Our company's existing expense report submission process is beset with difficulties such as insufficient information, erroneous data, and the inclusion of non-reimbursable charges, all of which cause delays, increased administrative workload, and employee aggravation. To overcome these issues, our project will create a "Expense Feedback System". This system will provide extensive, constructive comments on submitted expense reports, identifying particular issues such as missing entries, erroneous data, and non-reimbursable charges and providing clear instructions for correction.

The primary objectives of this project are to reduce error rates by minimizing incomplete or inaccurate reports through precise feedback, increase compliance by ensuring employees understand and adhere to company policies on reimbursable expenses, enhance efficiency by speeding up the approval process and reducing resubmissions, improve user experience by creating a more transparent and supportive process, and streamline financial processes by decreasing

administrative By deploying this system, we hope to create a more accurate, efficient, and user-friendly expense reporting procedure that will benefit both employees and the finance department.

1.4 SCOPE AND LIMITATIONS OF THE PROJECT

Users can utilize the model to provide input on their spending reports and receive recommendations for the adjustments needed to comply with corporate policies. In addition, it evaluates the accuracy of the data supplied and advises if the costs are eligible for reimbursement. However, due to restrictions in the training data or the model's restricted focus on particular spending categories, there can be mistakes in the feedback given. Additionally, the caliber and applicability of the user-provided data may have an impact on the model's efficacy. Therefore, before making any modifications to their cost reports, users should proceed with caution and double-check the model's suggestions.

The limitations of the project include the scarcity of real-time data provided to the model for training, thereby potentially impacting its ability to accurately assess the correctness of submitted values. Additionally, the project's classification system is narrowed down to specific classes, such as food, travel, and accommodation, with other expenses categorized under the generic "others" class. This limitation may lead to less granularity in expense categorization, potentially overlooking nuances in expense types and affecting the precision of the evaluation process.

CHAPTER 2

PROJECT ARCHITECTURE, DESIGN AND IMPLEMENTATION

2.1 SYSTEM ARCHITECTURE

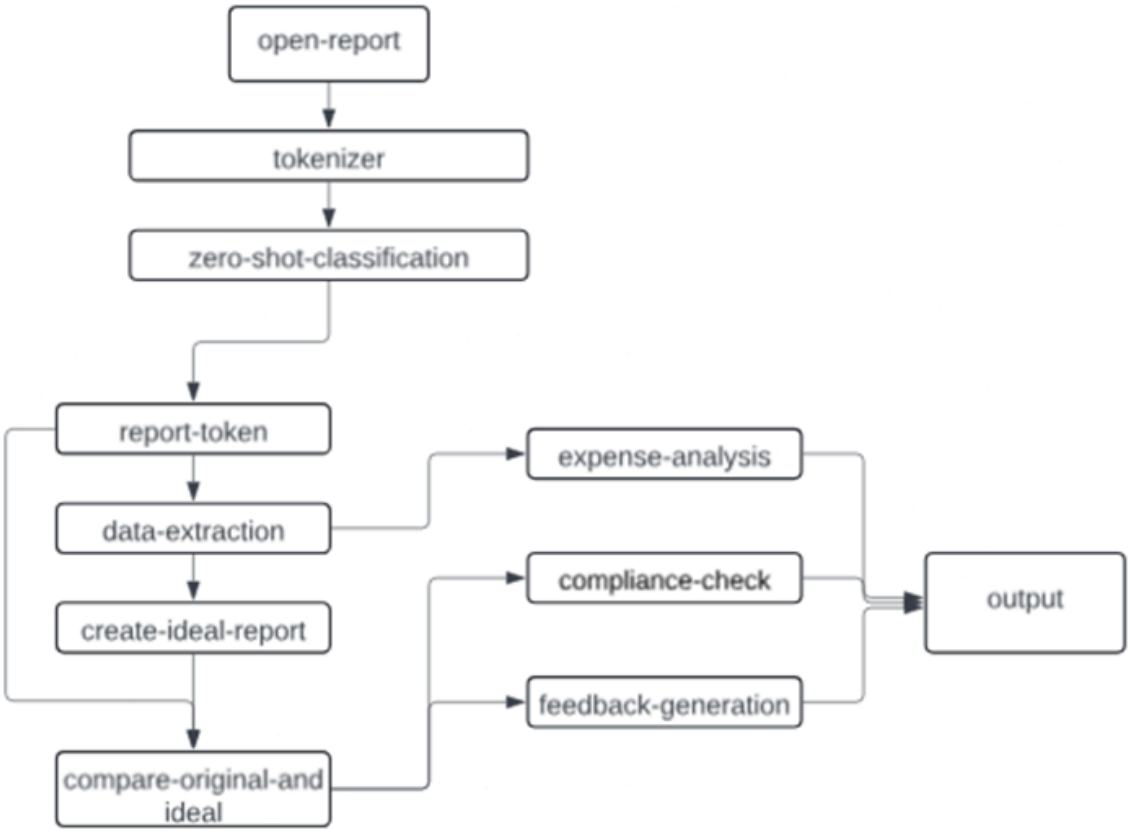

Fig 2.1.1 Architecture of expense feedback

The system architecture for creating a expense feedback using a fine-tuned model consists of following components:

The Expense Feedback project's system architecture is meticulously designed to deliver an efficient and reliable solution for evaluating and categorizing expense documents according to SAP standards. At its core, the architecture comprises a frontend layer built on ReactJS and a backend layer powered by Flask. ReactJS offers a user-friendly interface, enabling dynamic input of expense details, while Flask handles backend operations, such as document processing, API requests, and integration of machine learning models for classification.

Data collection and preparation are key components, sourcing from internal SAP data, synthetic data generated through tools like Gemini and ChatGPT, and real-time user data. This data undergoes rigorous preprocessing, including categorization, labeling, normalization, and text vectorization, ensuring it is suitable for training machine learning models.

Speaking of machine learning, the system leverages Support Vector Machine (SVM) and Bert Base Uncased models. SVM is employed for expense classification into predefined categories and determining their completeness and reimbursability, while Bert Base Uncased excels in advanced text classification, assessing the completeness of expense data by scrutinizing text tokens within expense categories.

Integration and communication between frontend and backend are seamless, facilitated by well-defined API endpoints, while testing and validation, utilizing tools like Postman, ensure system accuracy and performance metrics like precision,

recall, and F1-score are met. Deployment is robust, utilizing WSGI servers like Gunicorn to handle real-world traffic efficiently, with plans for continuous monitoring and updates to uphold system efficiency and compliance with SAP standards. Overall, this architecture promises a scalable, efficient, and compliant solution for expense management.

2.2 OVERVIEW OF THE DESIGN PROCESS

1. **Requirement Analysis:** Gathered requirements from stakeholders to understand the specific needs for evaluating expense documents, identifying necessary data fields, and defining classification categories in line with SAP standards.
2. **Frontend Development:** Chose ReactJS for its flexibility and robustness, focusing on creating an intuitive and interactive user interface for seamless expense detail input and compliance with SAP guidelines.
3. **Backend Development:** Selected Flask for its simplicity and scalability, designing the backend to handle API requests efficiently and ensure smooth data transfer and communication with the frontend.
4. **Data Collection and Preparation:** Created datasets using internal SAP data, synthetic data from tools like Gemini and ChatGPT, and real-time user data. Preprocessed data for normalization, text vectorization, and handling missing information.
5. **Model Training and Integration:** Trained a Support Vector Machine (SVM) model to categorize expenses and assess completeness and

reimbursability. Conducted hyperparameter tuning for optimization and integrated the model into the Flask backend.

6. **Implementation of Language Model:** Employed the Bert base uncased model for advanced text classification, using the transformers library to evaluate the completeness of expense data and identify valid tokens and outliers.
7. **Testing and Validation:** Rigorously tested the system using tools like Postman and performance metrics (precision, recall, F1-score), and refined the system based on user feedback from initial deployments.
8. **Deployment:** Deployed the application using WSGI servers like Gunicorn, establishing a production environment to handle real-world traffic, with continuous monitoring and updates planned for maintaining efficiency and accuracy.

2.3 EXPLANATION OF THE ENGINEERING PRINCIPLES USED IN THE DESIGN

ReactJS:

ReactJS is used for building user interfaces on the frontend of web applications. ReactJS is a powerful JavaScript library widely used for building dynamic user interfaces, particularly single-page applications (SPAs). It is known for its component-based architecture, which promotes reusability and modularity by breaking down the user interface into reusable components.

ReactJS also utilizes a virtual DOM for efficient rendering, improving the performance and responsiveness of the application. The component-based architecture facilitates code reusability and modularity. The virtual DOM enhances rendering performance. Its declarative syntax simplifies UI development and

improves code readability. React's rich ecosystem offers a variety of libraries and tools to expedite development, ensuring that it easily integrates with other technologies and scales well for complex applications.

Flask:

Flask is a lightweight Python web framework used to build scalable and maintainable web applications and APIs. It provides the necessary tools and libraries to develop robust backend services with minimal overhead. Flask's minimalist framework makes it easy to set up and use, offering flexibility in development. Its extensibility allows integration with various libraries and tools, while its micro-framework nature focuses on simplicity and speed, making it suitable for small to medium-sized applications.

Support Vector Machine:

The Support Vector Machine (SVM) algorithm is a key component in the classification tasks of this project due to its effectiveness in high-dimensional spaces and versatility in various classification problems. SVM excels when the number of features exceeds the number of samples and performs robustly even with noisy data or outliers. By maximizing the margin between different classes, SVM ensures better generalization on unseen data. Its flexibility is further enhanced through the use of different kernel functions, allowing it to handle both linear and non-linear separations efficiently. These characteristics make SVM an ideal choice for providing accurate and reliable classification in the Expense Feedback System, ensuring meaningful feedback for users.

Sklearn:

The SAP Expense Feedback System utilizes the scikit-learn library, a comprehensive and efficient Python library for machine learning, to implement the Support Vector Machine (SVM) algorithm for classifying expenses. Scikit-learn's consistent API, extensive documentation, and strong community support make it an ideal choice for this project. The library facilitates data preprocessing, model training, hyperparameter tuning, and performance evaluation. Key steps include using StandardScaler for feature scaling, SVC for creating and training the SVM model, and GridSearchCV for hyperparameter tuning. The trained model classifies new expense data, and the results are used to generate user feedback. Scikit-learn's integration with other Python libraries and its robust performance significantly enhance the system's capability to provide accurate and reliable feedback.

Datasets:

The datasets for training and testing were curated using Gemini and ChatGPT platforms. Gemini facilitated the generation of diverse and relevant expense-related data, ensuring comprehensive coverage of potential expense scenarios and variations. ChatGPT, on the other hand, was leveraged to enhance the conversational aspect of the dataset, enabling the creation of natural language interactions related to expense management. By combining the capabilities of Gemini and ChatGPT, the datasets were enriched with realistic expense entries and corresponding conversational contexts, providing a robust foundation for training the Supervised Learning Model (SLM). This approach ensured that the model was equipped to accurately classify expenses and generate insightful feedback based on the input data.

Hugging face:

Hugging Face serves as a pivotal resource in the development of small language models, offering a comprehensive suite of tools and pre-trained models through its Transformers library. Leveraging this platform, developers gain access to a diverse array of pre-trained models suitable for various tasks and deployment environments. The flexibility provided by Hugging Face extends to fine-tuning capabilities, enabling customization of models to specific use cases by training them on domain-specific datasets. Moreover, Hugging Face offers solutions for model compression, ensuring that smaller models can be deployed efficiently in resource-constrained environments without compromising performance. Additionally, Hugging Face provides seamless deployment options, facilitating the integration of small language models into applications and services across different platforms.

Small Language Model:

In the Expense Feedback project, the Bert base uncased model is utilized as a small language model to evaluate the completeness of expense data. The implementation involves a function that leverages the transformers library's pipeline function, specifically the text-classification pipeline with the Bert base uncased model. This setup enables the system to classify text tokens within expense categories effectively. The function processes the data by iterating over each category and its respective tokens. For each token, the classification pipeline is invoked to perform text classification. The results are interpreted to determine whether the token is valid or an outlier. Specifically, if the classification result's

label is 'LABEL_1', the token is deemed valid; otherwise, it is categorized as an outlier. The function compiles these results into a list of dictionaries, with each dictionary containing the token and its predicted label. This approach ensures a robust mechanism for assessing the completeness of data, aiding in the accurate classification and feedback generation for expense documents within the SAP framework. By leveraging the Bert base uncased model, the project benefits from advanced language processing capabilities, enhancing the precision and reliability of the expense evaluation process.

2.4 DESCRIPTION OF THE STEPS TAKEN TO IMPLEMENT THE PROJECT DESIGN

ReactJS:

The decision to employ ReactJS as the frontend framework for the Expense Feedback project stemmed from its renowned flexibility and robustness in crafting user interfaces. With ReactJS, the team embarked on creating an intuitive and interactive interface, facilitating seamless input of expense details for users. A paramount focus was placed on aligning the frontend design with SAP's stringent standards and guidelines for document evaluation, ensuring compliance at every step of the interface development process.

The frontend design of the Expense Feedback project incorporates a text bar allowing users to input their reports, with the system generating feedback based on whether the report is complete or incomplete. Additionally, the system produces a pie chart illustrating the distribution of complete and incomplete reports. In the

event of an incomplete report, the system provides insights into why it is deemed as such, often attributing it to missing data elements.

The inclusion of the text bar facilitates seamless interaction, enabling users to input their reports effortlessly. Through dynamic feedback generation, users receive prompt responses regarding the completeness of their submissions, enhancing user experience and workflow efficiency. The visualization of data completeness via a pie chart offers users a clear overview of their report status, aiding in decision-making and prioritization.

Back-end:

The backend development for the Expense Feedback project commenced with setting up the Flask framework, chosen for its simplicity and scalability. The initial step involved installing Flask and essential dependencies, followed by organizing the project directory for maintainability. The Flask application was initialized, incorporating blueprints to modularize the routes and streamline the handling of API requests. This setup facilitated efficient processing of expense documents and the generation of feedback.

The backend architecture was meticulously designed to ensure seamless integration with the ReactJS frontend, enabling smooth data transfer and communication between the two layers. A preprocessing script was developed to handle necessary data transformations, such as converting dates and normalizing text inputs. The trained classification model, essential for categorizing expenses, was integrated into the backend using joblib.

API endpoints were created to manage various functionalities, including the classification of expenses and the identification of missing data in incomplete reports. The backend routes processed incoming JSON data, utilized preprocessing functions, and generated predictions using the loaded model. Additionally, endpoints provided detailed feedback on incomplete reports, specifying which data elements were missing.

To ensure robust communication between the frontend and backend, CORS issues were addressed, and appropriate API endpoints were established. Testing was conducted rigorously using tools like Postman to validate the functionality of the API endpoints and ensure they handled various edge cases effectively. The final deployment of the Flask application was achieved using WSGI servers like Gunicorn, setting up a production environment that ensured the backend could handle real-world traffic efficiently. This comprehensive approach to backend development ensured a reliable and scalable system for processing expense documents and generating accurate feedback in line with SAP standards.

Support Vector Machine:

The Expense Feedback project incorporated a Support Vector Machine (SVM) for the classification of expense documents, leveraging its robust capabilities in handling high-dimensional data and producing accurate classifications. SVMs were chosen for their effectiveness in creating hyperplanes that optimally separate data into distinct categories. In this project, the SVM model was trained on a dataset that included various expense categories such as lodging, meals, travel, and other expenses, ensuring that it could accurately classify new expense entries into these predefined groups.

The process began with data collection, where relevant expense data was gathered and preprocessed to ensure consistency and accuracy. Features such as the amount, date, category, and description of expenses were used to train the SVM model. The preprocessing step included normalization and vectorization of textual data, converting it into a format suitable for the SVM classifier.

Once the model was trained, it was integrated into the Flask backend. When a new expense report was submitted, the backend utilized the SVM model to classify the expense into the appropriate category. This classification not only determined the category of the expense but also evaluated whether the expense adhered to SAP's reimbursement criteria. If the report was found incomplete, the system provided feedback on the missing data elements.

By employing SVM, the project benefited from a highly accurate and efficient classification system that improved the overall effectiveness of expense management procedures. The use of SVM ensured that the system could handle complex and diverse data inputs, providing precise and reliable feedback on expense documents.

Data Collection:

The dataset collection for the Expense Feedback project was a critical step in ensuring the accuracy and reliability of the Support Vector Machine (SVM) model used for classifying expenses and determining their completeness and reimbursability. The process began with leveraging existing SAP datasets, which included historical expense reports categorized and labeled according to SAP's reimbursement criteria. To fill in any gaps, synthetic data was generated using tools

like Gemini and ChatGPT, creating a comprehensive dataset that included various scenarios and expense types. Additionally, real-time data was collected from users during the initial deployment phases, refining the model to handle edge cases and unusual scenarios.

The dataset was meticulously divided into predefined categories such as lodging, meals, travel, and other expenses, with each entry containing relevant attributes like amount, date, category, and description. Expenses were labeled as complete or incomplete based on the presence of necessary data fields, and they were also tagged as reimbursable or non-reimbursable according to SAP's standards. This involved detailed annotations to indicate why certain expenses were non-reimbursable, such as policy violations or missing receipts.

Preprocessing steps included normalizing numeric fields to ensure uniformity and vectorizing textual data using techniques like TF-IDF or word embeddings. Missing data was handled through imputation for numeric values and default placeholders for text fields. The dataset was then split into training and test sets, typically in an 80-20 ratio, to validate the model's performance. Feature engineering was employed to enhance model accuracy, and techniques like oversampling or undersampling ensured the dataset was balanced with respect to different classes.

The preprocessed and labeled data was fed into the SVM model, which was trained to classify expenses accurately and assess their completeness and reimbursability. Hyperparameter tuning was conducted using methods like grid search to optimize

the SVM's performance. Finally, the trained model was validated against the test set, with metrics such as precision, recall, and F1-score used to assess its effectiveness. This comprehensive approach to data collection and preprocessing was crucial for the successful implementation of the Expense Feedback system, ensuring that the SVM model provided precise and reliable feedback on expense documents.

CHAPTER 3

RESULTS AND ANALYSIS

3.1 VALIDATION PROCEDURES

- The validation of giving a response on a request was done.
- Working functionality of buttons and other event listeners in UI were checked.
- A pipeline was created to generate feedback from the report test.
- Integration test of the pipeline with the frontend was done.

3.2 TEST RESULTS

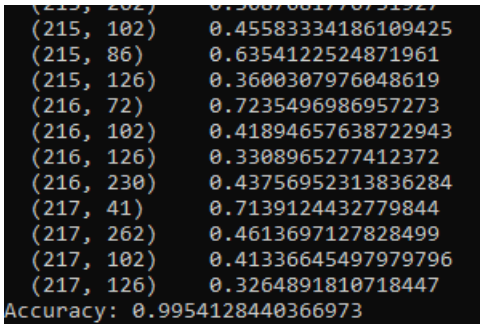

Fig: 3.2.1 : SVC Train Accuracy

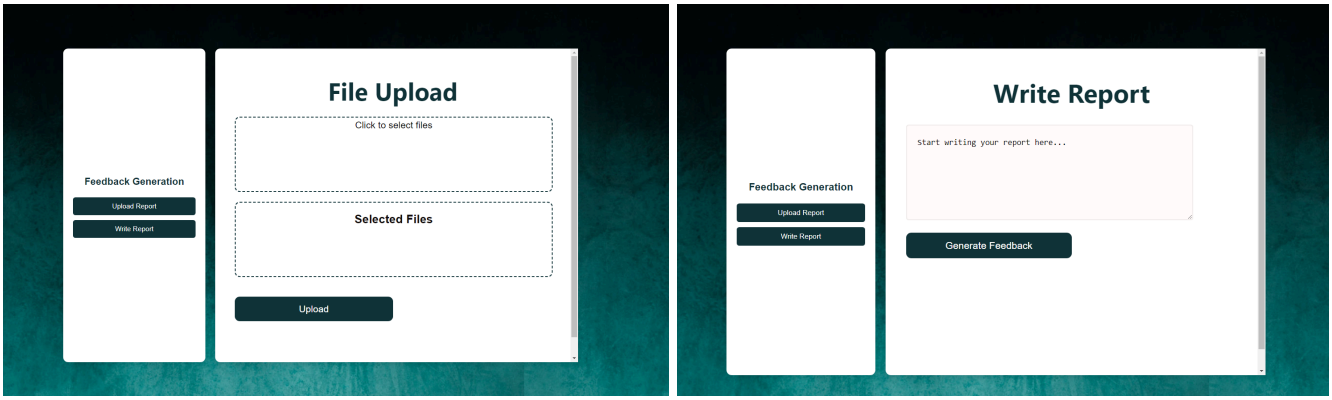

Fig 3.2.2 : Upload Report and Write Report UI

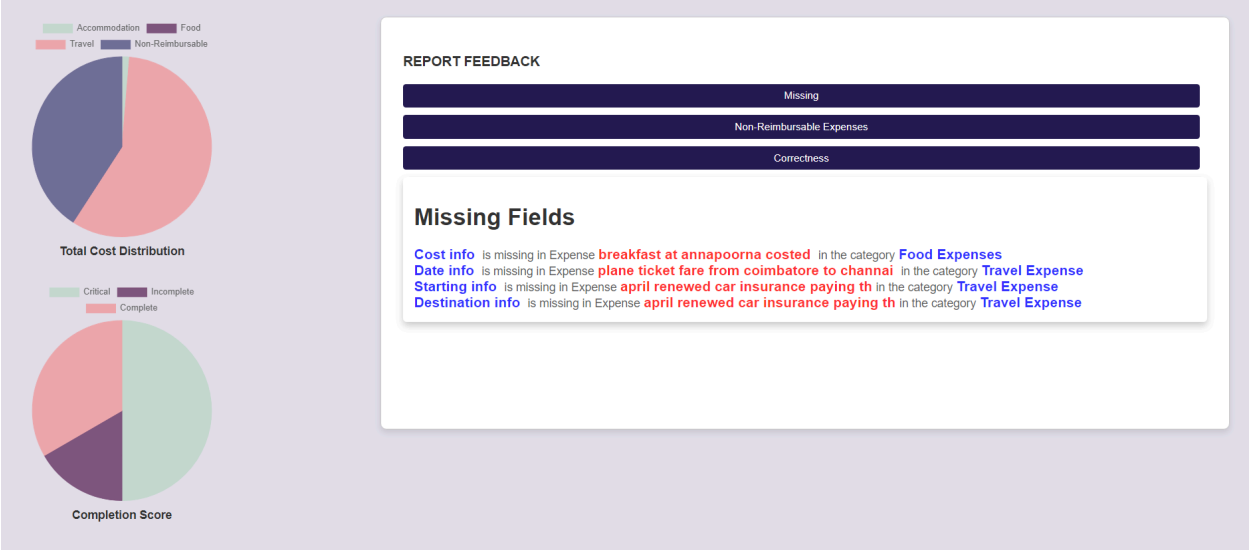

Fig 3.2.3 : Feedback Page

Fig 3.2.4 depicts the cost required in fine tuning the model

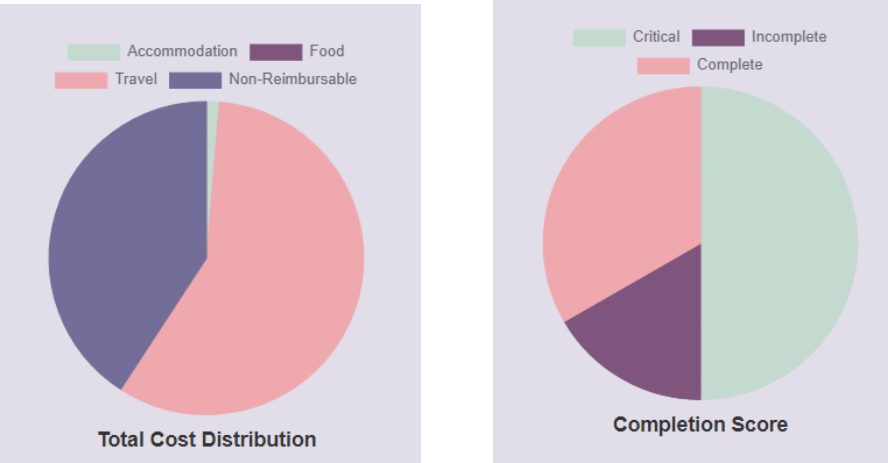

Fig 3.2.4 : Cost Distribution and Completion Score Charts

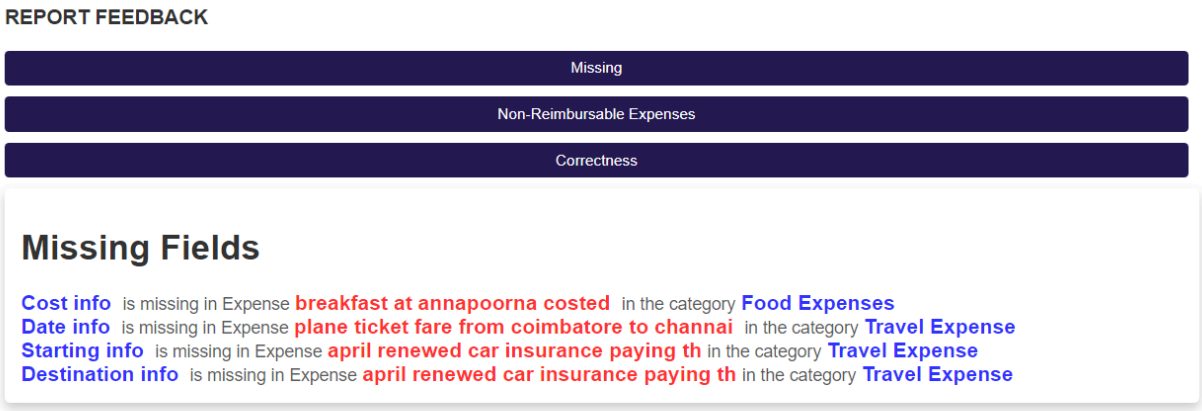

Fig 3.2.5 : Feedback on Missing values

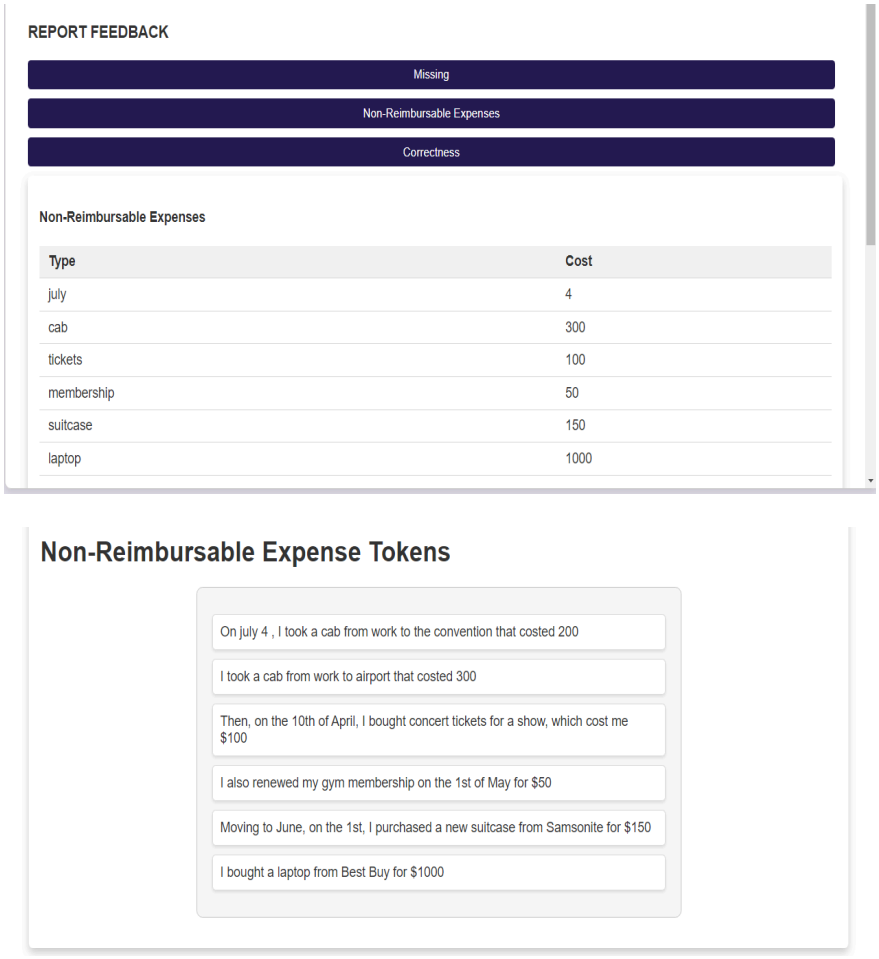

Fig 3.2.6 : Feedback on Non-Reimbursable expenses

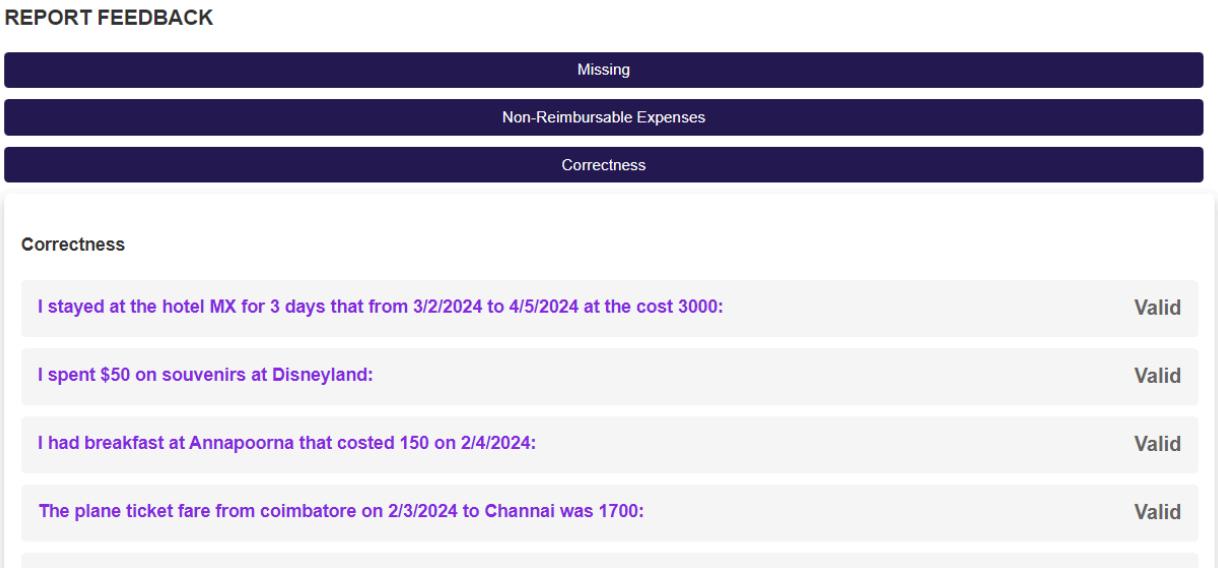

Fig 3.6 : Feedback on Validity of each expense

3.3 ANALYSIS OF RESULTS

Analyzing expense report feedback reports can be a treasure trove of insights for businesses. These reports often reveal common themes like missing receipts, inaccurate categorization, or delays in submissions. This can lead to inaccurate data, slow reimbursements, and potential policy violations. However, the benefits of analyzing this feedback are significant. Improved data quality, faster processing times, better policy compliance, and cost control are all achievable through addressing the issues identified in these reports. Companies can use this information to train employees, streamline reporting processes, adjust expense policies, and develop budgeting strategies. By taking these actions based on expense report feedback analysis, businesses can create a more efficient and cost-effective expense management system.

CHAPTER 4

LEARNING OUTCOMES

Understanding Natural Language Processing (NLP):Working on the Expense Feedback System project provided us with hands-on experience using NLP approaches. We learned how to parse expense descriptions, extract relevant keywords with NLTK, and use text classification models to estimate the plausibility of expenses. This helped us better grasp how NLP might be used in real-world jobs to improve data processing accuracy and efficiency.

Support Vector Classification (SVC) for Classification Tasks:During this assignment, we learned how to use Support Vector Classification (SVC) to classify expense report data. We learned how to train and evaluate SVC models to correctly identify and flag incomplete, incorrect, or non-reimbursable expense inputs, which improved the overall accuracy of the expense reporting process.

Using Transformers for Text Classification:We used the Hugging Face Transformers library to conduct text classification tasks with a BERT model. By establishing a text classification pipeline, we were able to assess the accuracy of expense-related statements and automate the feedback process. This required understanding how to fine-tune and apply pre-trained models to specific classification tasks.

Full-Stack Development:Creating the Expense Feedback System required both frontend and backend components. We obtained a thorough understanding of full-stack development by combining HTML, CSS, and JavaScript to create a user-friendly frontend interface, as well as using server-side logic for data

processing and feedback creation. This comprehensive strategy enabled us to deliver a seamless and efficient user experience.

API integration:API calls were necessary to integrate the BERT-based text classification model into our system. This experience taught us how to work with external APIs, manage authentication, and use the transformers library's functionality to improve our system's capabilities. We obtained real experience managing API requests and responses to ensure smooth operation.

Fine-tuning Models:To fine-tune the BERT model, we preprocessed data, trained it on specific tasks/datasets, and optimized its performance for our use case. This project provided us with insights into the fine-tuning process and the processes needed to adapt pre-trained models to specific applications. This experience strengthened our capacity to tailor machine learning models for specific performance enhancements.

By completing these learning objectives, we not only handled the immediate difficulties of expense report management, but also obtained essential skills in NLP, machine learning, full-stack development, and API integration that can be used for future projects and apps.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

The "Expense Feedback System" project solves major issues with the present expense report filing process, including inadequate information, erroneous data, and non-reimbursable charges. This solution attempts to reduce error rates, boost compliance with business regulations, improve process efficiency, user experience, and streamline financial operations by delivering detailed, constructive feedback. The introduction of this system will result in a more accurate, efficient, and user-friendly expenditure reporting procedure, benefiting both employees and the finance team. As we look ahead, new developments and integrations will continue to improve the system, ensuring that it remains a key tool for maintaining excellent financial management inside the organization.

To improve the "Expense Feedback System," numerous further advancements are planned. To begin, implementing advanced machine learning algorithms can aid in detecting and flagging probable errors or fraudulent activity in expense reports, thereby making the system more proactive. Expanding the system to incorporate a mobile application can boost employees' accessibility and convenience by allowing them to submit and analyze reports while on the road. Furthermore, providing a complete training program and user manuals based on typical feedback concerns can assist educate employees on recommended practices for expenditure reporting, thereby eliminating errors from the start. Finally, ongoing feedback from users will be required for iterative improvements, ensuring the system evolves to meet the changing needs of the company and its employees.

REFERENCES

- Liu, B., Bubeck, S., Eldan, R., Kulkarni, J., Li, Y., Nguyen, A., Ward, R., & Zhang, Y. (2023). TinyGSM: achieving >80% on GSM8k with small language models. *arXiv preprint arXiv:2312.09241*. https://doi.org/10.48550/arXiv.2312.09241
- Wongvorachan, T., & Bulut, O. (2022, December). The Open/Technology in Education, Society, and Scholarship Association Conference, 2(1). University of Alberta(Feedback Generation Through Artificial Intelligence)
- https://huggingface.co/docs/transformers/main/en/main_classes/pipelines
- https://scikit-learn.org/stable/modules/generated/sklearn.svm
- https://www.nltk.org/api
- https://aimresearch.co/uncategorized/small-language-models-how-slms-are-redefining.