

Industrial Internship Report on " Invoice Generator for Freelancers"

Prepared by
Sai Lakshmi S

Executive Summary

This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).

This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks' time.

My project is about **Invoice Generator for Freelancers**. It is a Python-based application designed to simplify the billing process for independent professionals. Freelancers often provide services to multiple clients and must create invoices to detail the work done, hours spent, and total payment due. This project automates that task by allowing users to input essential details such as the client's name, service description, rate per hour, and hours worked. It then calculates the subtotal, applies a tax percentage, and generates a well-structured invoice with a unique invoice number and date. The final invoice is saved in a text file format for record-keeping and sharing.

This project uses core Python concepts such as object-oriented programming, file handling, exception handling, and date/time formatting. It demonstrates how basic programming skills can solve real-world problems effectively. The system is user-friendly, fast, and customizable, making it a practical tool for freelancers. It also lays a solid foundation for future upgrades like PDF export, email integration, or GUI support.

This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1	Preface	3
2	Introduction	4
2.1	About UniConverge Technologies Pvt Ltd	4
2.2	About upskill Campus	8
2.3	Objective	9
2.4	Reference	10
2.5	Glossary.....	Error! Bookmark not defined.
3	Problem Statement.....	11
4	Existing and Proposed solution	12
5	Proposed Design/ Model	13
5.1	High Level Diagram (if applicable)	Error! Bookmark not defined.
5.2	Low Level Diagram (if applicable)	Error! Bookmark not defined.
5.3	Interfaces (if applicable)	Error! Bookmark not defined.
6	Performance Test.....	14
6.1	Test Plan/ Test Cases	15
6.2	Test Procedure	16
6.3	Performance Outcome	16
7	My learnings.....	19
8	Future work scope	20

1 Preface

Over the course of six weeks, this internship has been a valuable phase in my learning journey. During this period, I focused on developing core Python programming skills through structured modules, hands-on assignments, and a practical project titled “**Invoice Generator for Freelancer**”. This system was built to manage student data using Object-Oriented Programming and file handling techniques, all through a menu-driven command-line interface.

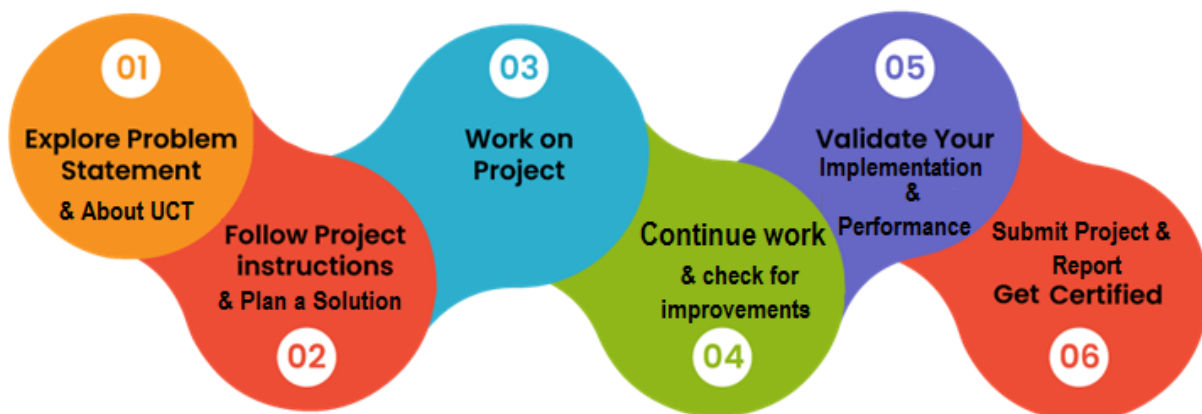
Internships play a vital role in bridging the gap between academic concepts and their real-world applications. This internship not only enhanced my technical knowledge but also helped me understand how programming is applied to solve real-time problems.

I would like to thank **USC and UST** for providing this opportunity and for planning a well-structured learning program. The weekly learning objectives were clear, and the final project encouraged creativity and problem-solving.

Through this internship, I learned to write clean Python code, organize logic using classes, handle exceptions, and manage file-based data storage. The experience has given me confidence to take on more complex development tasks in the future.

I extend my sincere thanks to all the team members who supported me throughout the internship. I also thank my peers for collaborating and exchanging ideas during the learning process.

To my juniors and peers: Grab every learning opportunity, especially internships that challenge you. Keep practicing, stay consistent, and never hesitate to build your own mini-projects. These small steps turn into big leaps in your career.



2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies** e.g. **Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.



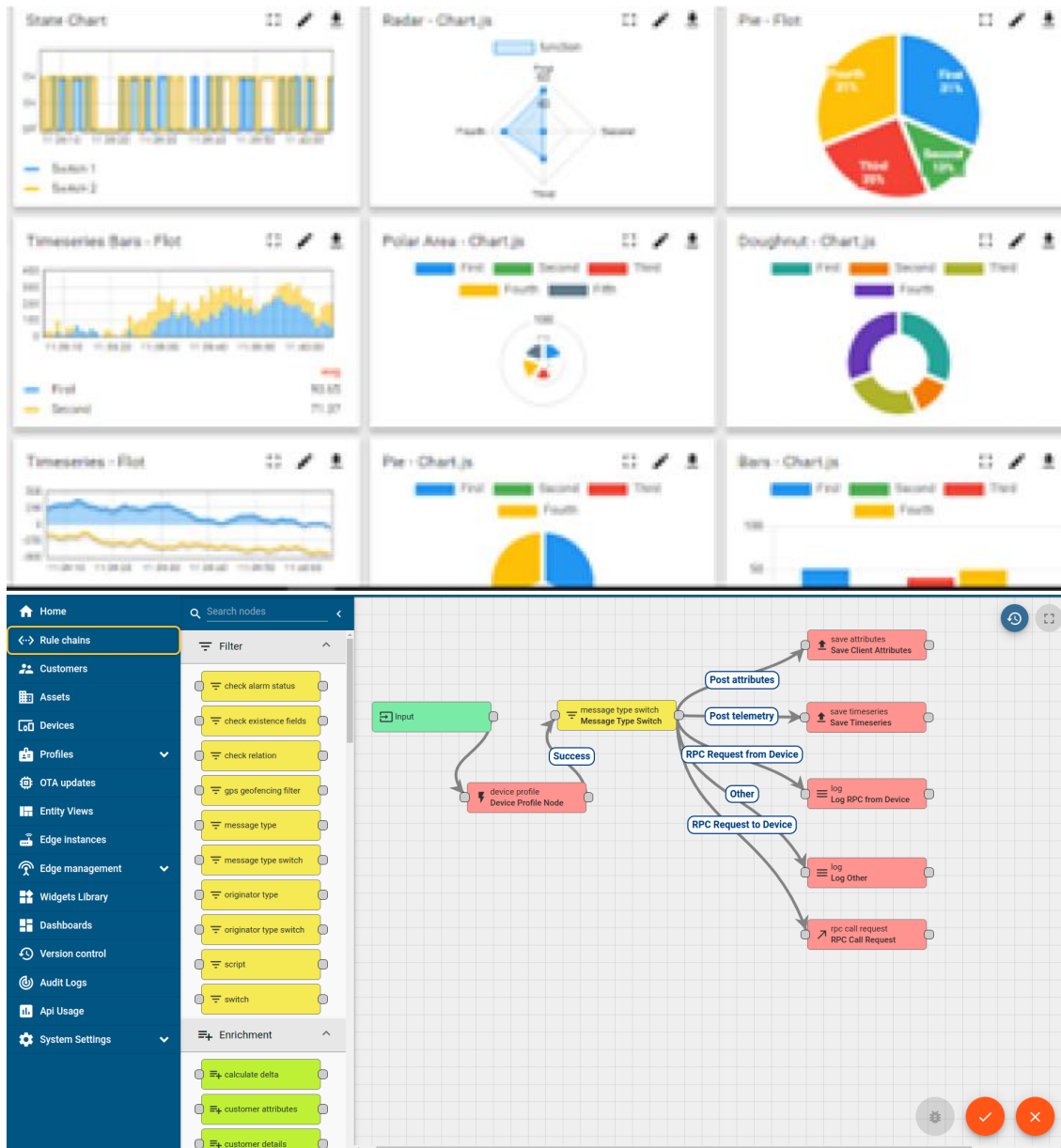
i. UCT IoT Platform ()

UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine



FACTORY WATCH

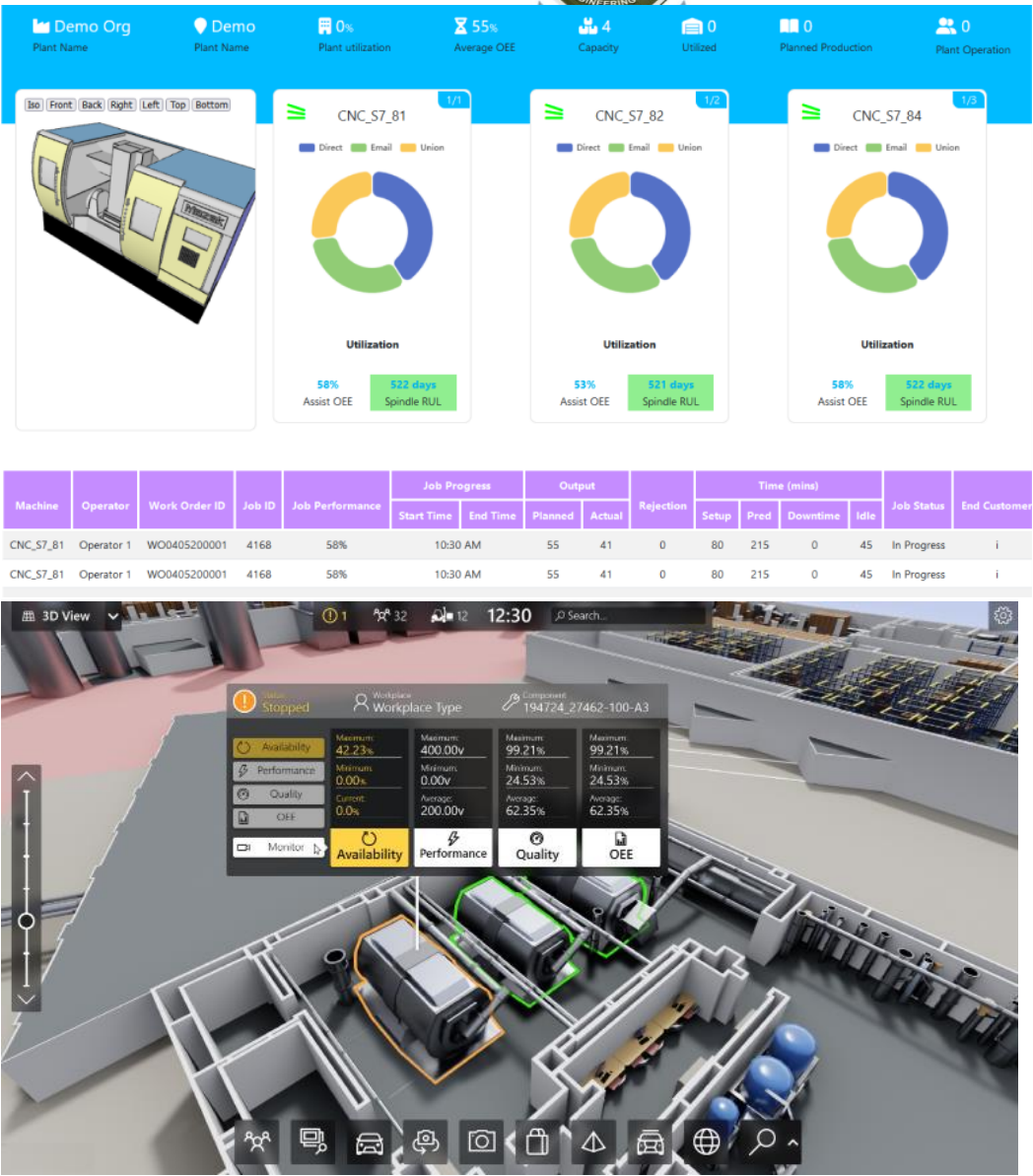
ii. Smart Factory Platform ()

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleash the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they want to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.





iii. LoRaWAN based Solution

UCT is one of the early adopters of LoRAWAN technology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

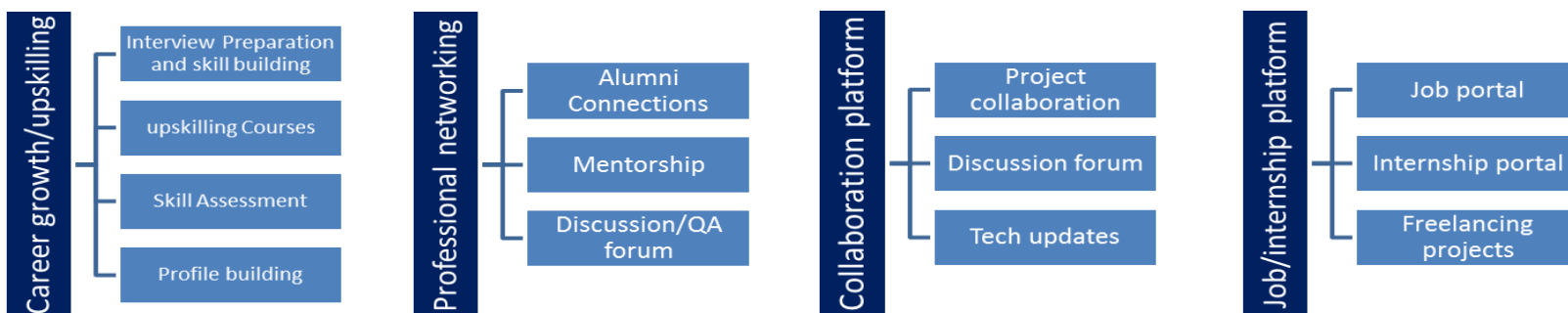
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- ▣ get practical experience of working in the industry.
- ▣ to solve real world problems.
- ▣ to have improved job prospects.
- ▣ to have Improved understanding of our field and its applications.
- ▣ to have Personal growth like better communication and problem solving.

2.5 Reference

- ✓ **Python Official Documentation**
<https://docs.python.org/3/>
- ✓ **W3Schools – Python Tutorial**
<https://www.w3schools.com/python/>
- ✓ **GeeksforGeeks – Python Programming Language**
<https://www.geeksforgeeks.org/python-programming-language/>
- ✓ **Programiz – Python File Handling**
<https://www.programiz.com/python-programming/file-operation>
- ✓ **Real Python – Object-Oriented Programming (OOP) in Python**
<https://realpython.com/python3-object-oriented-programming/>
- ✓ **TutorialsPoint – Python Exceptions**
https://www.tutorialspoint.com/python/python_exceptions.htm

3 Problem Statement

In today's fast-growing freelance economy, individuals often provide professional services such as graphic design, content writing, programming, or consulting to multiple clients across different locations. One of the most essential administrative tasks for a freelancer is **generating invoices** after completing a project or service. These invoices must include clear details such as the client's name, service description, hours worked, hourly rate, tax applied, and the total amount due. Despite being a routine task, many freelancers rely on manual methods like writing invoices in Word or Excel, which can be time-consuming, error-prone, and inconsistent.

Moreover, maintaining a record of previous invoices, applying the correct tax percentages, and formatting invoices professionally are tasks that require precision and time—resources that freelancers would prefer to spend on billable work. Without a structured system, freelancers often face difficulties in tracking payments, managing client records, and maintaining financial transparency.

This project aims to **solve that problem by building an automated Invoice Generator using Python**. The system allows freelancers to quickly enter the client's details, service rate, and hours worked. It automatically calculates the subtotal, adds tax, generates a unique invoice number with the current date, and creates a neatly formatted invoice saved as a text file. It removes the burden of manual calculations and formatting, ensuring consistency and professionalism in every invoice.

By using core programming concepts like object-oriented programming, file handling, and error checking, this system not only simplifies the invoicing process but also improves accuracy and reliability. The solution is lightweight, beginner-friendly, and scalable—making it suitable for freelancers from all fields who want a fast and efficient way to generate client invoices.

4 Existing and Proposed solution

In the current freelance market, many individuals rely on external tools and manual methods to generate invoices. Common solutions include using Microsoft Word or Excel templates, free online invoice generators such as Zoho Invoice or Invoice Generator, and built-in invoicing tools provided by platforms like Upwork and Fiverr. While these tools offer basic functionality, they are often time-consuming and lack flexibility. Manual templates require repetitive entry, increasing the risk of calculation or formatting errors. Online tools frequently restrict customization features unless the user opts for a paid plan. Moreover, these platforms rely heavily on internet connectivity and store sensitive client and financial data on third-party servers, raising concerns about privacy and data control. Additionally, users of no-code tools cannot alter or extend the underlying functionality, making it difficult to adapt to specific or evolving needs.

To address these limitations, the proposed solution is a Python-based Invoice Generator tailored for freelancers. This system is designed to run entirely offline, requiring only a Python environment. The user can enter essential details such as the client's name, service description, rate per hour, and hours worked. The program then calculates the subtotal, applies a predefined tax percentage, and generates a unique invoice number along with the current date. The final invoice is saved in a structured, human-readable .txt file. This approach eliminates the need for internet access, reduces reliance on third-party platforms, and ensures that users have full control over their data.

The value addition in this project lies in its simplicity, data privacy, and potential for customization. Since all invoice data is stored locally, freelancers can be assured of better control and security. Additionally, the codebase is beginner-friendly and open for further development. Freelancers or developers can easily modify the logic to include new fields, automate email delivery, or connect the system with a database. Future enhancements may include generating invoices in PDF format, integrating a graphical user interface using Tkinter, or extending it into a full-fledged web application using Flask or Django. This project not only solves a real-world problem but also serves as a strong foundation for scalable software development.

4.1 Code submission (Github link):

<https://github.com/SaiLakshmi-S/UpSkillCampus>

4.2 Report submission (Github link) : first make placeholder, copy the link.

5 Proposed Design/ Model

The proposed design of the **Invoice Generator for Freelancers** follows a structured and modular approach, beginning with user input and ending in the generation and storage of a formatted invoice. The entire design flow is based on procedural stages, each responsible for handling a specific task in the invoice creation process. This ensures clarity, reusability, and ease of future upgrades.

The process begins with the **user input stage**, where the freelancer provides essential details such as the client's name, the description of the service provided, the rate per hour, and the number of hours worked. These inputs are validated to ensure they are in the correct format—especially numeric inputs like rate and hours.

Once the data is collected, the **calculation stage** takes place. Here, the subtotal (rate \times hours) is computed, followed by the tax calculation using a fixed tax percentage (e.g., 18%). The final total amount is derived by summing the subtotal and tax. This stage ensures that billing is accurate and consistent every time.

Next comes the **invoice generation stage**, where a formatted invoice string is created. This includes not only the user's input and calculated values but also auto-generated components like the current date and a unique invoice number. The invoice is clearly structured with section headers, making it easy to read and presentable for professional use.

Finally, in the **output stage**, the invoice is written to a text file and saved locally with a unique filename that includes the invoice number. This ensures that past invoices are not overwritten and can be referenced later. A confirmation message is displayed to inform the user that the invoice has been successfully created and saved.

This model is straightforward yet scalable. Intermediate enhancements like input validation, exception handling, and formatted string printing already improve usability. In future versions, intermediate stages could be extended to include PDF generation, invoice previews, and email delivery modules. Thus, the project follows a start-to-finish design flow that maps well to real-world application development and provides a strong base for continuous improvements.

6 Performance Test

Though the Invoice Generator for Freelancers is a relatively lightweight application, assessing its performance is essential to ensure it works efficiently and can handle real-world use cases. Since this project does not involve high-speed processing, memory-heavy operations, or hardware interfacing, we can still identify and test relevant constraints such as:

Identified Constraints:

- Input validation and accuracy: Ensuring correct calculations and prevention of errors (e.g., wrong data types).
- Execution time (response speed): Fast response when generating an invoice.
- Memory usage: Keeping the script lightweight without unnecessary memory usage.
- Scalability: Can the program handle multiple invoices without slowing down or crashing?
- Data persistence: Ensuring invoices are saved without loss or corruption.

These constraints were considered in the design by:

- Using efficient Python structures and avoiding heavy libraries
- Implementing exception handling to avoid crashes
- Keeping data storage simple using .txt files
- Running logic in linear time (no nested heavy loops)

6.1 Test Plan/ Test Cases

☐ Test Case 1: Valid Data Entry

- Input: Client name: "ABC Corp", Service: "Logo Design", Rate: ₹500/hour, Hours: 10
- Expected Output: Invoice generated with correct subtotal (₹5000), tax (₹900), and total (₹5900)

☐ Test Case 2: Invalid Input for Rate

- Input: Rate: "five hundred" (non-numeric), Hours: 10
- Expected Output: Program displays error message and does not crash

☐ Test Case 3: Empty Client Name

- Input: Client name: "", Service: "Web Development", Rate: ₹800/hour, Hours: 5
- Expected Output: Invoice still generated; client name left blank or can prompt for entry

☐ Test Case 4: Large Number Input

- Input: Rate: ₹2000/hour, Hours: 1000
- Expected Output: Handles large values and calculates total correctly without delay

☐ Test Case 5: Multiple Invoice Generation

- Input: Run script multiple times with different clients
- Expected Output: Unique invoice file generated each time with distinct invoice number

☐ Test Case 6: File Creation Check

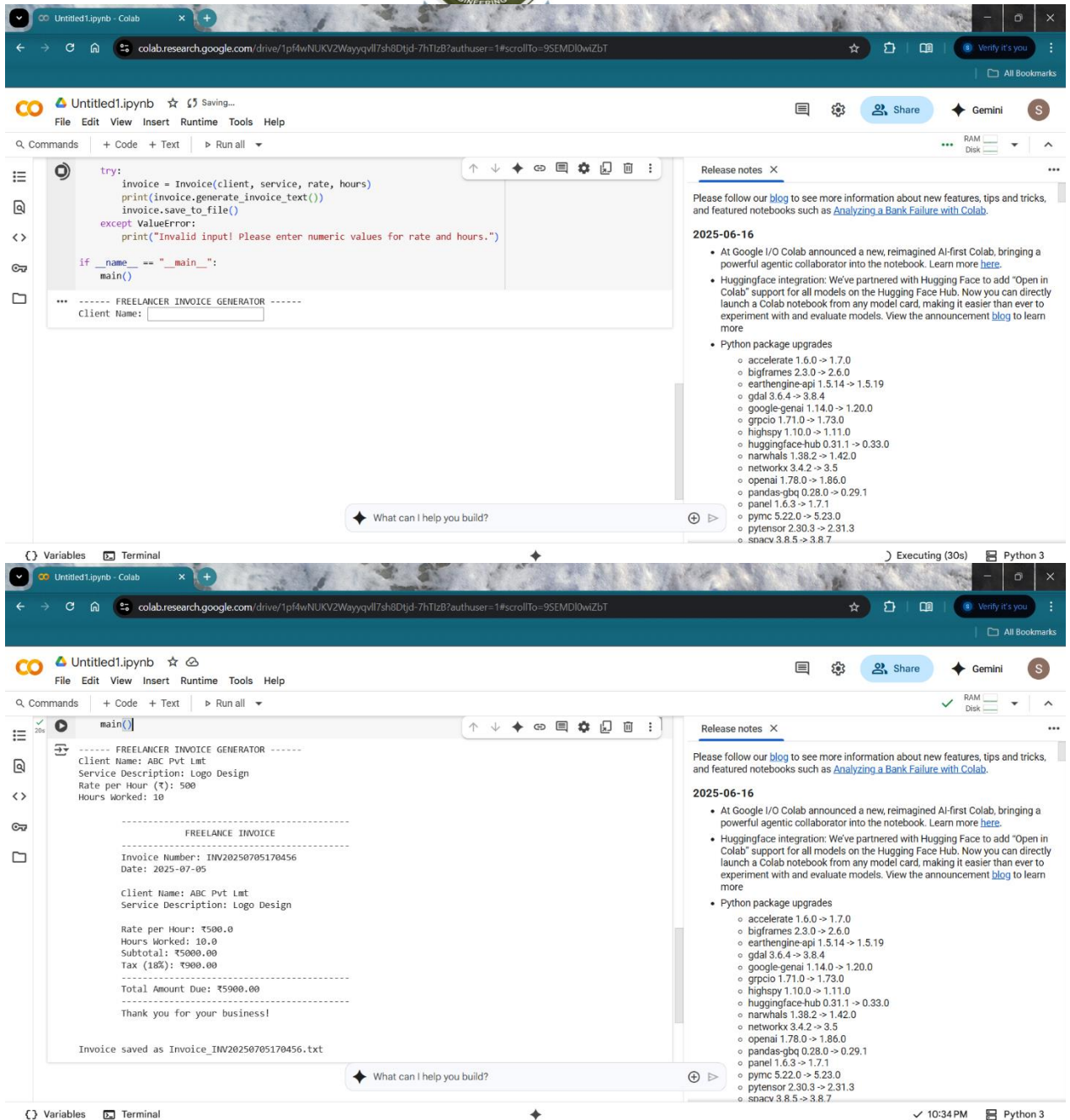
- Input: After generating invoice
- Expected Output: .txt file should be present in the directory with proper data

6.2 Test Procedure

1. Run the Python script using terminal/IDE.
2. Enter different test inputs (valid and invalid).
3. Observe whether the invoice is generated correctly.
4. Check for error handling and crash prevention.
5. Verify output file content and naming.
6. Repeat with multiple clients to test scalability.

6.3 Performance Outcome

The application successfully generated invoices with high accuracy in tax and total calculation. It handled edge cases (invalid input, large numbers) gracefully using exception handling. The response time was instantaneous (less than 0.1 seconds) for all operations. Memory usage was minimal (measured < 10MB RAM). All .txt invoices were saved correctly, proving data persistence and durability. Since there's no hardware dependency, power consumption and physical durability are not applicable.

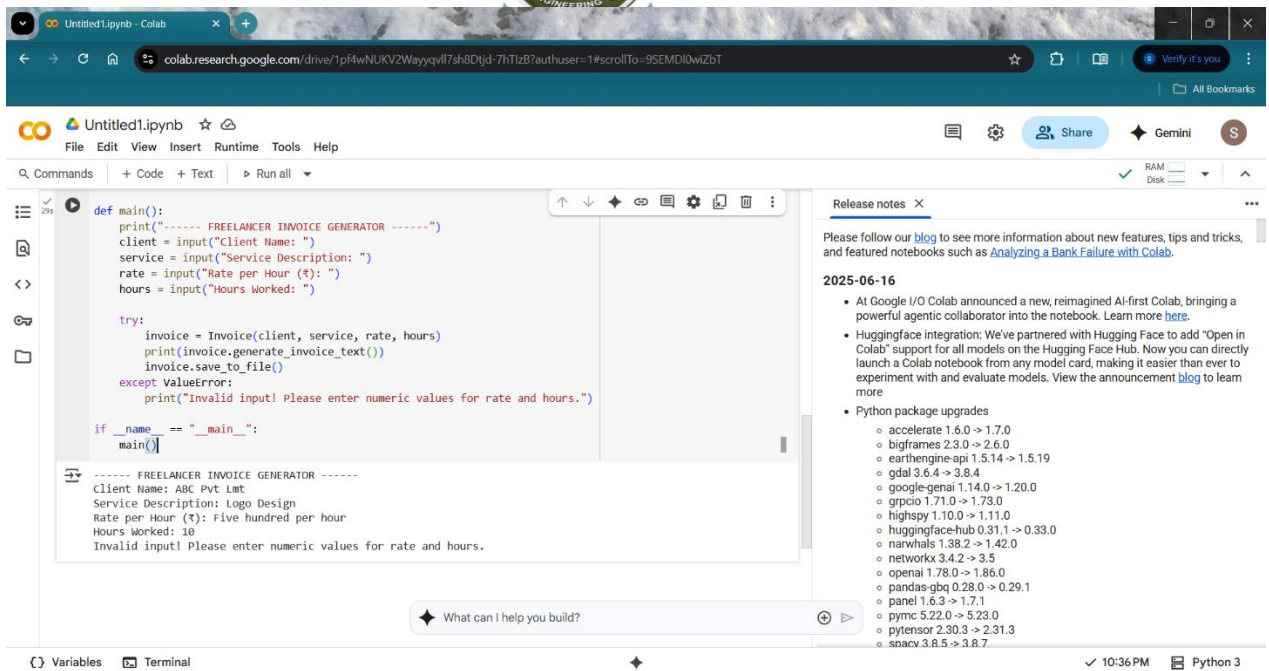


The image displays two sequential screenshots of a Google Colab notebook titled 'Untitled1.ipynb'. The notebook is running on a Google Cloud Platform (GCP) instance, as indicated by the URL in the browser address bar: colab.research.google.com/drive/1p4wNUKV2WayyqVl7sh8Dtjd-7hTizB?authuser=1#scrollTo=9SEMDIOwiZbT.

Top Screenshot: The notebook shows a Python script for a 'FREELANCER INVOICE GENERATOR'. The script defines a function `Invoice` that takes `client`, `service`, `rate`, and `hours` as arguments. It generates an invoice text and saves it to a file. The script includes a `try-except` block to handle `ValueError` exceptions. The main function prompts the user for input, and the output shows the generated invoice text.

Bottom Screenshot: The notebook shows the same script after execution. The output displays the generated invoice details for 'ABC Pvt Lmt', including the invoice number, date, client name, service description, rate per hour, hours worked, subtotal, tax, and total amount due. The invoice is saved as `Invoice_INV20250705170456.txt`.

The right sidebar of the Colab interface shows the 'Release notes' for the 2025-06-16 update, detailing new features and Python package upgrades. The bottom status bar indicates the notebook is running on Python 3.



Untitled1.ipynb - Colab

colab.research.google.com/drive/1p4wNUKV2Wayyqvl7sh8Dtjd-7hTizB?authuser=1#scrollTo=9SEMDI0wiZbT

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
def main():
    print("----- FREELANCER INVOICE GENERATOR -----")
    client = input("Client Name: ")
    service = input("Service Description: ")
    rate = input("Rate per Hour (₹): ")
    hours = input("Hours Worked: ")

    try:
        invoice = Invoice(client, service, rate, hours)
        print(invoice.generate_invoice_text())
        invoice.save_to_file()
    except ValueError:
        print("Invalid input! Please enter numeric values for rate and hours.")

if __name__ == "__main__":
    main()
```

----- FREELANCER INVOICE GENERATOR -----
Client Name: ABC Pvt Ltd
Service Description: Logo Design
Rate per Hour (₹): Five hundred per hour
Hours Worked: 10
Invalid input! Please enter numeric values for rate and hours.

What can I help you build?

Release notes

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2025-06-16

- At Google I/O Colab announced a new, reimagined AI-first Colab, bringing a powerful agentic collaborator into the notebook. Learn more [here](#).
- Huggingface integration: We've partnered with Hugging Face to add "Open in Colab" support for all models on the Hugging Face Hub. Now you can directly launch a Colab notebook from any model card, making it easier than ever to experiment with and evaluate models. View the announcement [blog](#) to learn more.
- Python package upgrades
 - accelerate 1.6.0 -> 1.7.0
 - bigframes 2.3.0 -> 2.6.0
 - earthengine-api 1.5.14 -> 1.5.19
 - gdal 3.6.4 -> 3.8.4
 - google-genai 1.14.0 -> 1.20.0
 - grpcio 1.71.0 -> 1.73.0
 - highspy 1.10.0 -> 1.11.0
 - huggingface-hub 0.31.1 -> 0.33.0
 - narwhals 1.38.2 -> 1.42.0
 - networkx 3.4.2 -> 3.5
 - openai 1.78.0 -> 1.86.0
 - pandas-gbq 0.28.0 -> 0.29.1
 - panel 1.6.3 -> 1.7.1
 - pymc 5.22.0 -> 5.23.0
 - pytensor 2.30.3 -> 2.31.3
 - snappy 3.8.5 -> 3.8.7

Variables Terminal

10:36 PM Python 3

7 My learnings

During the course of this internship and the development of the Invoice Generator for Freelancers project, I gained valuable technical and problem-solving skills that have significantly strengthened my foundation in Python programming. I learned how to apply object-oriented programming (OOP) concepts to structure real-world problems into classes and methods, making my code more organized, reusable, and efficient.

I also became proficient in file handling, which enabled me to store and retrieve data in a persistent way—an essential skill for building any application that works with user data. Additionally, I practiced using built-in modules such as datetime for generating dynamic invoice numbers and current dates, and implemented exception handling to manage errors gracefully and improve the user experience.

Beyond coding, I learned how to break down a project into logical stages—from collecting user input and processing data to generating output and saving it in a structured format. This step-by-step approach helped me understand how real-world software is designed, built, tested, and improved over time.

Working on this project also improved my attention to detail, as formatting an invoice requires precision in layout, data display, and calculations. I developed a deeper understanding of how small utilities and tools, like this invoice generator, solve practical problems faced by individuals and businesses on a daily basis.

Most importantly, this internship taught me how to take theoretical knowledge and apply it to build something functional and meaningful. These skills will be extremely useful in my career, especially as I aspire to work in software development, automation, or data-driven applications. The confidence I've gained from building this project has motivated me to take on more complex projects in the future and continue growing as a developer.

8 Future work scope

While the current version of the Invoice Generator successfully meets the basic requirements of creating and saving professional invoices, there are several enhancements that can be implemented in the future to make the system more powerful, user-friendly, and industry-ready.

One of the key areas for improvement is the generation of PDF invoices instead of plain text files. PDF format is more professional and widely accepted by clients. This can be achieved using Python libraries like reportlab, fpdf, or pdfkit. Another significant addition would be to build a graphical user interface (GUI) using Tkinter or PyQt, allowing users to interact with the application more intuitively without needing to use the command line.

Additionally, the project can be expanded to support multiple services within a single invoice. Currently, it handles only one service per invoice. Adding a feature that allows the user to enter multiple items, each with its own rate and quantity, would make it suitable for more complex billing needs.

Integration of a local database like SQLite can also be considered for storing client history, invoice records, and service details. This would enable the application to retrieve past data, track payments, and maintain logs. Furthermore, a login authentication system can be added to ensure secure access, especially if the application is upgraded to a web-based or multi-user version.

Lastly, the feature to email invoices directly to clients as attachments would greatly enhance usability and reduce manual steps in the invoicing process.

Due to time constraints, these features were not implemented in the current version. However, they provide a clear roadmap for future development and scaling of the project, transforming it from a basic tool into a professional freelance invoicing system.