Progress Report 1

Topic: Visual Drilling Operations & Predictive Maintenance System

Student Name: Uchechi Sebastian

Student ID: 300393092

Course: CSIS 4495 002 Applied Research Project

Section: 002

Team Lead: Uchechi Sebastian (Individual Project)

Table of Contents

Work Date / Hours Log

Date	Hours	Description
17-Jan-2026	3.5	- I explored and researched possible project ideas for the course, focusing on areas related to drilling operations and data analytics. I evaluated whether the project idea was feasible to implement within the course timeline and researched whether similar work had been done before. This helped me narrow down and decide on a suitable project that aligns with my background and course requirements.
20-Jan-2026	3	- I started learning React by reviewing the basics of components, JSX, and how a React application is structured. I focused on understanding how React will be used to build the frontend of my applied project.
21-Jan-2026	3.5	- I worked on Introduction of the project: Improving the introduction, stating the problem statement clearly, outlining what questions needs to be answered. - I searched for literature and research work based on my project to find similar research on my project. I searched for gaps within this research project work.
22-Jan-2026	2.5	- I continued literature research related to drilling operations, nonproductive time, and decision support systems. I reviewed how similar systems visualize operational data and noted limitations in existing approaches, particularly depth-based visualization. - I worked on my reference and included it in my project report after the research I did.
23-Jan-2026	3	- I worked on defining the project methodology, including data sources, data collection approach, and analysis techniques. - I also clarified how simulated drilling data would be used and documented the justification for using non-proprietary datasets in the project.
25-Jan-2026	1	- I created the project timeline for the complete project for the entire semester.

26-Jan-2026	3.5	- I drew the Gantt chart for the project timeline and included it in my proposal report. - I prepared the appendix. - I went through the entire report and did a lot of corrections and restructuring to make sure the proposal fits what my project is all about.
27-Jan-2026	2	- I set up the project GitHub repository structure according to the course requirements and organized folders for implementation and documentation. - I created the frontend, backend, and data directories under the implementation folder to support a clean development workflow.
29-Jan-2026	5	- I initialized the React frontend using Vite within the repository and verified that the development server runs correctly on my local machine. This step confirmed that the frontend environment was properly configured and ready for further development. - I continued learning React by implementing the initial application layout and navigation structure. I worked on creating reusable components for the application frame (layout and navigation) and began organizing the project into pages and components to follow best practices. I separated the HTML structure (JSX) and styling (CSS) into different files to improve readability and maintainability. I also installed and configured React Router to support page navigation between the wells list, dashboard, and reports sections of the application. - I implemented the well selection page and routing logic to allow navigation between different wells and their corresponding dashboard views. I tested navigation behavior to ensure that selecting a well correctly loads the appropriate dashboard page. During this process, I encountered and resolved common React

		development issues, including file naming inconsistencies, incorrect import paths, and component casing errors. I fixed these issues by standardizing file names, correcting relative import paths, and restarting the development server as needed, ensuring the application builds and runs successfully after all fixes.
02-Feb-2026	3.5	- -Built and refined Wells page with color-coded well status cards - Implemented depth-based wellbore visualization - Added clickable depth segments with event details modal - Integrated routing between wells and dashboards - Improved frontend layout and interaction flow
05-Feb-2026	4	- I set up flexible column normalization so the upload can handle different Excel header names - I added validation for wells, operations, and events during the upload process - I added row-level error reporting to make debugging easier and catch bad data - I improved the duplicate checks so existing records aren't inserted - I added logging across the upload flow for better visibility - I finished wiring up the /upload endpoint with multipart file support - I updated the utils for mapping, normalization, and safer data parsing - I prepared the backend for future analytics by making sure uploaded data is clean and consistent

06-Feb-2026	2	- I created simulated datasets and collected sample datasets from different companies so I could compare how each one structures its drilling data. - I analyzed the differences in column names, formats, units, timestamps, and category labels across these datasets.
09-Feb-2026	5	- Implemented a complete rewrite of the Excel upload pipeline for wells, operations, and events. - Fixed all column-name mismatches between the transformer output and SQLAlchemy models. - Updated the Well insertion logic to use well_name instead of the invalid name argument. - Corrected the operations insertion to use operation_type instead of the non-existent op_type. - Updated event insertion and duplicate-check logic to use the correct model field (event_time). - I rebuilt the transform_dataset() function to ensure consistent normalization, depth conversion, timestamp parsing, and category mapping. - Ensured all transformed sheets contain the required columns before insertion. - Improved error reporting for wells, operations, and events to make debugging easier. - Cleaned up duplicate imports and removed unused or inconsistent logic. - Stabilized the entire ETL flow so uploads now run end-to-end with predictable behavior. -

Summary Description of work done during this reporting period

This reporting period focused on moving from proposal planning into Week 4 implementation. I finalized the project direction and validated feasibility by researching drilling operations decision support, dashboard limitations, and reusable software expectations. I then set up the project repository structure and initialized the frontend using React (Vite), confirming the application runs locally.

I started implementing the Week 4 deliverables: main layout and navigation foundation, well selection screen, and preparation for a per-well dashboard view. Key issues encountered were mostly development setup problems typical for beginners (React routing installation timing, file-name casing mismatches such as Layout vs layout, and incorrect relative import paths).

I resolved these by standardizing file naming conventions, correcting imports, restarting the Vite server after changes, and organizing the UI into reusable components with separate CSS files. No major changes to the project proposal scope were made, but the UI implementation plan is now clearer and mapped to weekly milestones.

I continued working on the frontend development of the project using React. I implemented and refined the Wells page by enhancing the visual design of the well cards using color-coded styling to clearly differentiate well statuses (Normal, Warning, and Critical). This improves usability by allowing users to quickly identify high-risk wells at a glance.
I also worked on the drilling dashboard interface, focusing on the depth-based wellbore visualization and interaction flow. I connected the well selection page to individual well dashboards using React Router and ensured that selecting a well correctly loads its corresponding dashboard data.
Additionally, I integrated interactive behavior between the wellbore visualization and the event details modal, allowing users to click on depth segments and view contextual drilling event information. I resolved React state-handling and component-communication issues encountered during this process and verified that the application runs correctly after the updates.

I set up the initial FastAPI backend with SQLite, project structure, virtual environment, and database models for wells, operations, and events.

I set up flexible column normalization so the upload can handle different Excel header names, and I added validation for wells, operations, and events during the upload process. I also added row-level error reporting to make debugging easier and catch bad data. On top of that, I improved the duplicate checks so existing records aren't re-inserted, and I added logging across the upload flow for better visibility. I finished wiring up the /upload endpoint with multipart file support, updated the utils for mapping, normalization, and safer data parsing, and prepared the backend for future analytics by making sure uploaded data is clean and consistent.

I created simulated datasets and collected sample datasets from different companies so I could compare how each one structures its drilling data. I analyzed the differences in column names, formats, units, timestamps, and category labels across these datasets.

I focused on stabilizing the entire Excel-upload workflow by aligning the transformation pipeline, the upload endpoint, and the SQLAlchemy models. I corrected several mismatches between transformed column names and model fields, including well_name vs name, operation_type vs op_type, and event_time vs timestamp. The transform_dataset() function was rewritten cleanly to ensure consistent normalization, depth conversion, date/timestamp parsing, and category mapping. The upload route was updated to use the correct model field names, fix duplicate-check logic, and prevent keyword-argument errors during inserts. With these fixes, the ETL pipeline now processes wells, operations, and events reliably and returns accurate insert/skip/error counts.

Repo Check in of Implementation completed

I have checked in my work to the GitHub repository according to the course requirements. The repository currently includes the following folders and files:

1. **ReportsAndDocuments/**
    - **Proposal/**
        - Project proposal document.
    - **ProgressReports/**
        - Progress Report #1
2. **Implementation/**
    - **frontend/**
        - Initialized React frontend project using Vite
        - src/ directory containing:
            - pages/
                - Wells
                - Dashboard
                - Reports
                - Maintenance
            - components/
                - KpiCard
                - Layout
                - SegmentModal
                - Wellbore
            - styles/ (separate CSS files for pages and components)
        - package.json and related configuration files
    - **backend/**
        - app/
            - __init__.py
            - main.py
        - models/
            - __init__.py
            - well.py
            - operation.py
            - event.py
        - routers/
            - __init__.py
            - wells.py
            - operations.py
            - events.py
            - analytics.py
            - risk.py
            - upload.py

        - schemas/
            - well_schema.py
            - operation_schema.py
            - event_schema.py
        - services/
        - utils/
            - __init__.py
            - column_mapping.py
            - transform.py
            - database.py
    - **data/**
        - Folder created to store simulated or example datasets for later phases

3. Miscellaneous - Misc
4. **README.md**

AI Use Section

AI use Table

AI Tool	Version / Account	Specific Use	Value Added
ChatGPT	GPT-5.2	Assisted with refining the project idea, structuring the proposal, improving academic writing, and clarifying the research design, methodology, and timeline.	Helped organize thoughts clearly, improve academic clarity, and ensure the project scope and structure align with the course.
Copilot	Free	Summary of Relevant Literature and Existing Research	Helped to clearly identify the existing research on the project and showed how the project differs from various working projects.

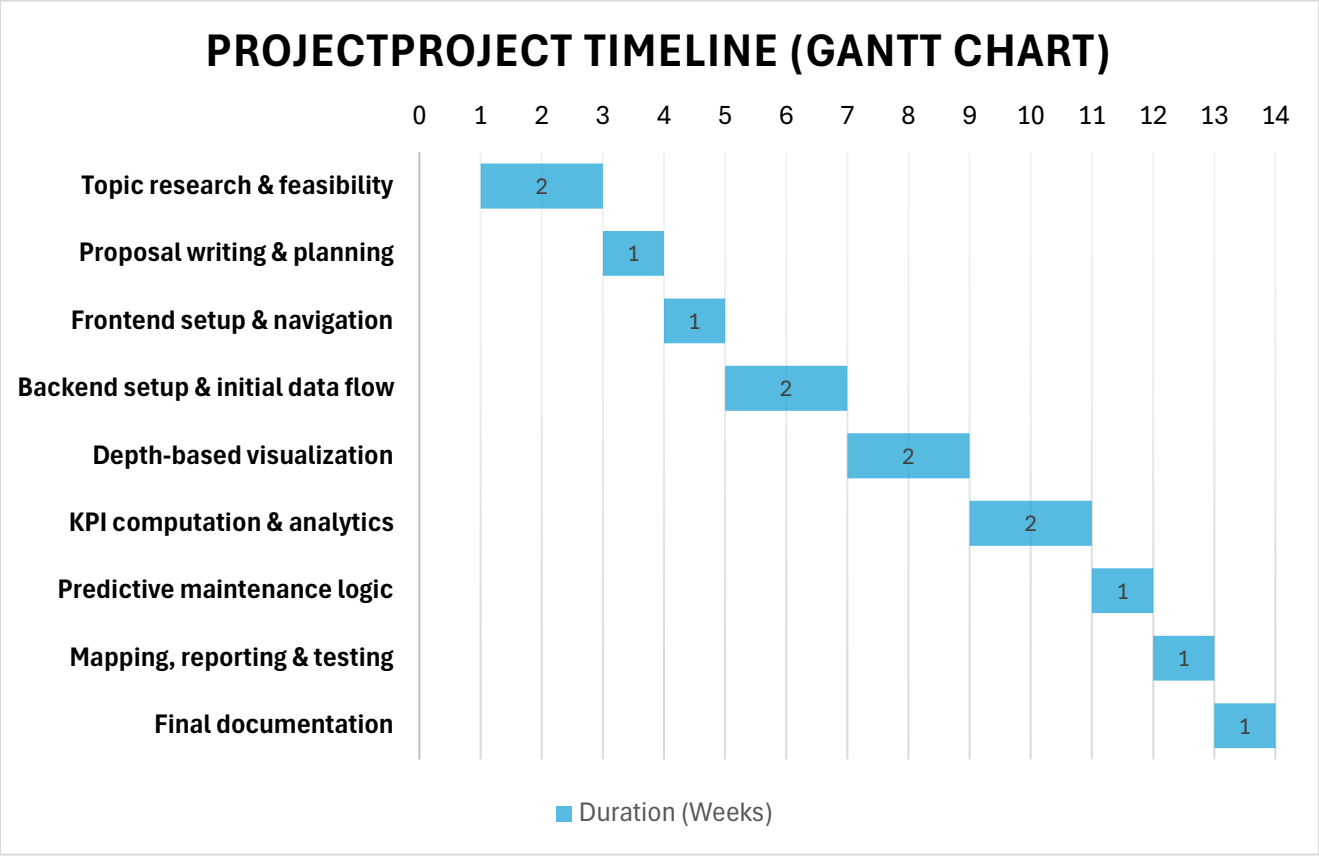
ChatGPT	GPT-5.2	Used as learning support while developing the React frontend, including understanding React components, JSX structure, routing with React Router, file organization, and troubleshooting common setup and import issues during implementation.	Supported faster understanding of React concepts, helped resolve configuration and import errors, and improved confidence in implementing the frontend correctly within the project timeline.
Figma AI	Free	Used to generate UI layout suggestions and design inspiration for the drilling dashboard, including card layout, spacing, and visual hierarchy.	Helped to clearly identify the existing research on the project and showed how the project differs from various working projects.
Copilot (Backend Development)	Free	Assisted with setting up the FastAPI backend, configuring SQLite, creating database models, organizing folder structure, and planning the data-upload workflow.	Accelerated backend setup, ensured clean architecture, and improved clarity on how data flows from upload → database → analytics → frontend.
Copilot (Backend Development)	Free	I worked through the Excel upload pipeline, including column normalization, validation, duplicate handling, row-level error reporting, and debugging the upload endpoint.	It helped me break down the errors clearly, understand what was causing the failures in the upload pipeline, and refine the normalization and validation flow so the backend is more stable, predictable, and better prepared for the analytics features I'll build next.

Project Planning and Timeline

Project Timeline

Phase	Week(s)	Milestones	Deliverables
Phase 1: Topic Exploration & Domain Research	Weeks 1 - 2	Explored potential project domains Reviewed drilling operations context Identified problem area and feasibility	Project idea selection Initial domain understanding
Phase 2: Proposal Development & Planning	Week 3	Defined project scope and objectives Identified data requirements Prepared a formal project proposal	Submitted project proposal Finalized research questions and methodology
Phase 3: Frontend Development (Core UI)	Week 4	Implement the main application layout Build well selection and navigation views	Initial React frontend Well selection interface
Phase 4: Backend Setup & Initial Data Flow	Weeks 5–6	Set up backend framework (FastAPI) Designed initial database schema Implemented basic API endpoints Connected frontend to backend using sample data	Functional backend service Initial API endpoints Frontend consuming backend data
Phase 5: Depth-Based Visualization	Weeks 7–8	Implemented vertical wellbore (pipe) visualization Mapped drilling events to depth intervals	Interactive depth-based visualization
Phase 6: KPI Computation & Analytics	Weeks 9–10	Compute NPT and event-based KPIs Aggregate data by depth and operation	KPI analytics module
Phase 7: Predictive Maintenance Logic	Weeks 11	Implement rule-based risk indicators Link risk to recorded events and equipment	Predictive maintenance indicators with explanations
Phase 8: Mapping & Report Generation & Testing	Weeks 12	Implement a well location map view Enable downloadable drilling summary reports. Test system functionality.	Location-based well view Report export feature Tested application
Phase 9: Final documentation	Week 13	Prepare final documentation	Final project report

Gantt chat – Project timeline



Appendix

AI Prompt(s) Used

- Explain React components, pages, layout, and routing like I'm a beginner, and guide me step-by-step to set up a React app using Vite on Windows.
- I'm building a drilling dashboard UI. Help me create a Week 4 React layout with wells list + navigation. Keep CSS in separate files and explain what each file does.
- I'm getting an import/casing error in Vite/React. Diagnose the error and tell me exactly what files to rename and what import paths to use.
- How do I modify the frontend such that when I click on the view detailed explanation, it goes ahead to show me the detailed explanation and the analytics gotten based on the data provided? Please also explain very well so I can understand how it is done.
- No there is already <button className="primaryBtn" type="button"> View Detailed Explanation </button> <button className="secondaryBtn" type="button" onClick={onClose}> Close </button> so when you chick on the button that's where the three pictures come in.
- okay but when I put the backend it will generate the analytics on its own without putting any explanations

Microsoft Copilot

- let us work based on my project. do I need the data to start so that I can do some analytics to connect it to the front end? or let us start with , obviously you upload your dataset, then the app should receive this dataset and analyze it and show it on the wells page. When you then click on the particular well, it will show you the drilling pipe with different color coding. But first we should work on uploading the dataset first.
- For the uploading of files, these 3 categories, are you uploading the three documents as one csv, that is the; wells, operations, events. Also, most of this dataset have to be prepared after drilling right? that means there has to be some data input into their various tables. Or can the dataset from the company have everything, then the app begins to separate it by itself. That is to say, does it have to be prepared already.

- lets continue the upload i think most companies use excel not csv right so is it better to use excel to upload the data or csv based on what companies use.
- I dont want the exact names because not everyone in the industry would have time to put the exact names.
- okay if I integrate the script into the endpoint upload, that means it is ready for use. what does that mean. Also, after integrating it, can i test it to see if it is working. secondly will the upload be able to carry thousands of roles of data