PROJECT PLAN

Cloud Cost Intelligence Platform

Michael, Ishan, Sean, Bryana, Tony

CMSC 495 Computer Science Capstone

Instructor: Lynda Metallo

January 20, 2026

Table of Contents

1. OVERVIEW

This team project serves as a capstone demonstration of full-stack development, data aggregation, and dashboard design skills.

This project plan describes a Cloud Cost Intelligence Platform. It covers the statement of work, project scope, requirements, case scenarios, team roles, milestones, schedule, and risk management. The plan also explains how team members will share information.

The Cloud Cost Intelligence Platform helps organizations that use more than one cloud provider improve cost visibility and control. Many companies with services like AWS and Azure spend 30-40% more than needed because of unused resources, limited visibility, and scattered billing data.

Our solution offers a single dashboard that brings together cost data from different cloud providers, reviews spending habits, finds waste, and gives clear suggestions for saving money. It is designed for IT managers, DevOps teams, and finance departments who manage cloud budgets.

2. PROJECT DELIVERABLES

The Cloud Cost Intelligence Platform helps users track and manage their cloud spending with a web interface. The dashboard collects cost data from providers like AWS and Azure. It shows spending trends, highlights unused resources, and suggests ways to save money. Our team will build the platform and deliver it as a web app, then present it to the instructor and stakeholders.

Deliverable	Description	Due Date
Project Plan	This document; defines scope and schedule	Jan 20, 2026
Project Design	Architecture, data models, UI mockups	Jan 27, 2026
Phase I Source Code	Core backend, database, basic frontend	Feb 3, 2026
Peer Review I	Individual evaluation of team contributions	Feb 3, 2026
Test Plan	Test cases for unit, integration, system testing	Feb 10, 2026
Phase II Source Code	Dashboard, analytics, recommendations	Feb 17, 2026
User Guide	End-user documentation	Feb 24, 2026
Peer Review II	Final individual evaluation of team contributions	Mar 3, 2026
Final Report	Complete project documentation	Mar 3, 2026

3. PROJECT SCOPE

3.1 In Scope

End State: User opens dashboard → sees AWS + Azure costs → views trends → gets flagged on waste → sees recommendations → exports report.

Key Functions:

- Web-based application
- Cost data aggregation from AWS and Azure
  - Simulated
  - Actual (time permitting)
- Spending trend visualizations (charts, graphs)
- Waste identification
  - Unused resources
  - Idle resources
- Basic recommendations:
  - Hard-coded logic
  - Intelligent (additional dependency requirements)
- Budget threshold alerts
- Resource usage metrics (global average or max by timespan)
- Exportable cost reports
  - CSV
  - PDF

3.2 Scope Boundaries

Category	Key Functions	Time Permitting	Out of Scope
Cloud Data	Simulated	Live API integration	Real time streaming
Providers	AWS + Azure	GCP	Beyond 3 providers
Recommendations	Hard code rules	Intelligent/ML	Automatic/ Chat bot
Export	CSV + PDF	Direct email	Chat integration
Authentication	None (demo)	Basic login	Multi-user
Alerts	User-defined thresholds	Email notifications	Text notifications

3.3 Firm Out of Scope

Key out of scope functions include Mobile application, purchasing functionality, productional deployment.

4. REQUIREMENTS

4.1 Business Requirements

- Provide users a clear view of cloud cost
- Help users find ways to save on cloud spending
- Let users track budgets and get alerts when spending nears editable limits
- Create reports that users can export and share with stakeholders
- Show users how their spending over time changes, forecasting
- Give users resource usage data to help them make capacity decisions.

4.2 Functional Requirements

ID	Requirement	Description
FR-01	View Cost Dashboard	Display total costs by provider on single screen
FR-02	View Spending Trends	Show costs over time with daily/weekly/monthly charts
FR-03	Filter by Provider/Service	Drill down by AWS vs Azure or by service type
FR-04	View Waste Alerts	Flag resources that are unused or underutilized
FR-05	View Recommendations	Show rightsizing suggestions with potential savings
FR-06	Export Reports	Download cost data as CSV or PDF
FR-07	Set Budget Thresholds	Configure spending limits with user-friendly alert thresholds. Users set custom warning levels based on their business requirements.
FR-08	View Resource Metrics	Display global average or global max resource usage. User can select metric type and date range.

4.3 Technical Requirements

Category	Requirement
Operating System	Cross-platform (Windows, Mac, Linux via web browser)
Frontend	React (HTML/CSS/JS)
Backend	Python
Database	MS SQL Server, SQL Server Management Studio
Browser Support	Chrome, Firefox, Edge

5. CASE SCENARIO - USER WALKTHROUGH

5.1 Scenario: IT Manager Reviews Monthly Costs

An IT Manager wants to review cloud spending and identify cost-saving opportunities.

1. [FR-01] - User opens the web application to view the dashboard (home screen)
2. [FR-01] - Dashboard shows the total monthly cost (e.g., AWS $12,450 & Azure $8,230).
3. [FR-03] - User filters by "AWS" to focus on specific providers.
4. [FR-02] - User clicks "View Trends" to view chart showing spending over past 30 days.
5. [FR-02] - User can notice spikes usage by day and click the event to view more details.
6. [FR-02] - System shows volume of new instances "EC2" created on that day.
7. [FR-04] - User opens "Waste Alerts" and finds flags for days with low usage.
8. [FR-07] - User sets a monthly budget and enables alerts.
9. [FR-06] - User clicks "Export Report" to download a CSV.

5.2 Scenario: DevOps Engineer Finds Unused Resources

A DevOps Engineer wants to identify and clean up unused cloud resources.

1. [FR-01] - User opens web application and navigates to "Waste Alerts" from dashboard.
2. [FR-04] - System displays a list of resources with low utilization.
3. [FR-04] - Resource entry includes name, type, provider, monthly cost, and utilization.
4. [FR-03] - User filters "Azure" to focus on specific providers.
5. [FR-08] – User selects a timespan/ views global average CPU utilization for instances.
6. [FR-04] - User sees instances with very low utilization with cost.
7. [FR-05] - User selects "View Recommendations" to see options for action
8. [FR-05] - System suggest downgrading "Large" to "Small" and displays savings.
9. [FR-07] – User sets a custom alert threshold for their team budget, not a system default.
10. [FR-07] – System confirms threshold saved and will trigger alerts.

5.3 Scenario: Finance User Generates Monthly Report

A Finance User needs to pull cost data for the monthly budget review.

1. [FR-01] - User opens the application, and the Dashboard screen appears.
2. [FR-01] - Dashboard displays total cost for each Cloud service.
3. [FR-03] - User filters by date range
4. [FR-06] - User clicks "Export Report" and downloads a PDF.

5.4 Error Handling Scenarios

1. [FR-01] - Data fails to load: "Unable to retrieve data. Please try again."
2. [FR-03] - No data exists for selected date range: "No data available for {date range}"
3. [FR-06] - Export fails: "Export failed. Please check your connection and try again."
4. [FR-07] - Invalid threshold value entered: "Please enter a valid number."
5. [FR-08] - No usage data for selected timespan: "No resources for {date range}."

6. TEAM MEMBER ROLES

Everyone on the team helps write code for the project. Leads are responsible for their area's deliverables, coordinate tasks, and act as the main contact, but they are not the only ones contributing. Team members help leads as needed, depending on sprint priorities and who is available.

Project Manager

Assigned To: Michael
Organizes team meetings and keeps track of progress so that all deliverables meet requirements and deadlines. Acts as the main contact for any issues or blockers. Handles communication across different time zones (PST, EST, CET) and brings together team input for final submissions. Watches scope boundaries set in Section 3 and reviews feature requests. Reachable on Teams or by phone.

Frontend Developer Lead

Assigned To: Ishan
Develops the user interface with React (HTML, CSS, and JavaScript). Main tasks are linked to the functional requirements below:
FR-01: Dashboard view displaying total costs by provider (AWS, Azure)
FR-02: Trend charts showing daily/weekly/monthly spending patterns
FR-03: Filter controls for provider, service type, and date range
FR-04: Waste alerts table with resource name, type, cost, and utilization
FR-05: Recommendations display with savings estimates
FR-06: Export buttons triggering CSV and PDF downloads
FR-07: Threshold input form for user-defined budget alerts
FR-08: Metric selector (average/max) and date range picker
Makes sure the design works well on all devices and browsers (Chrome, Firefox, Edge).
Connects frontend components to backend API endpoints.

Backend Developer Lead

Assigned To: Sean
Creates the REST API and business logic in Python. Main tasks are matched to the functional requirements below:
FR-01: Endpoint returning aggregated cost data by provider
FR-02: Endpoint returning time-series spending data for trend charts
FR-03: Query filtering logic for provider, service, and date range
FR-04: Waste detection algorithm identifying unused/underutilized resources
FR-05: Recommendation engine generating rightsizing suggestions with savings calculations
FR-06: Report generation service producing CSV and PDF exports
FR-07: Threshold storage and alert trigger logic
FR-08: Usage metrics calculation (global average or max for selected timespan)
Manages data coming in from simulated cloud providers. Documents all API endpoints so the frontend team can use them.

Database Developer Lead

Assigned To: Tony

Designs and builds the database structure with MS SQL Server. Main responsibilities include:
Schema Design: Tables for providers, resources, cost_records, usage_metrics, alerts, recommendations, and user_thresholds
Data Relationships: Foreign keys linking resources to providers, costs to resources
Queries: Aggregation queries for dashboard totals, trend data, and waste identification
Simulated Data: Seeds database with realistic AWS/Azure cost and usage data for demo
Performance: Indexes for common query patterns (by provider, date range, resource type)
Keeps data accurate and supports all functional requirements by making data easy to retrieve.

Tester / QA Lead

Assigned To: Bryana
Leads the testing approach and quality checks. Main responsibilities include:
Test Cases: Creates test cases for all 8 functional requirements (FR-01 through FR-08)
Scenario Validation: Tests user walkthroughs from Section 5 (IT Manager, DevOps, Finance)
Error Handling: Validates all 5 error scenarios from Section 5.4
Edge Cases: Identifies boundary conditions (empty data, invalid inputs, large datasets)
Integration Testing: Verifies frontend-backend-database communication
Documentation: Records test cases and results in Test Plan deliverable
Note: Each team member tests their own code. Bryana oversees the overall QA strategy and does the final checks.

Documentation

Handles all project documentation, including:
User Guide: Step-by-step instructions for end users with screenshots
Final Report: Complete project documentation including lessons learned
README Files: Setup instructions for GitHub repository
API Documentation: Endpoint reference (coordinated with Sean)
Formatting: Keeps a consistent style across all deliverables
Organizes document review cycles and makes sure everything is submitted on time.

7. RESOURCE LIST

7.1 Team Members

Name	Time Zone	Primary Skills	Availability
Ishan	EST	AWS, Azure, Python, LaTeX, React	-After 1200 weekdays -After 1500 Sat/Sun
Michael	PST	Python/Django, Java, Docs	-1000 to 1200, After 2000 M-F -All day Sat/Sun
Bryana	EST	Java, SQL, Organization	- Outside of business hours
Sean	EST	Dev, JS, Python	-After 1800 EST weekdays -All day Sat/Sun
Tony	CET	Palantir, Data Pipelines, Python, SQL	1300-1600 EST weekdays -All day Sat/Sun

7.2 Technical Resources

Resource	Details
Development Machines	Personal computers (Windows/Mac/Linux)
Version Control	GitHub (repository: CMSC495-CloudCost)
Frontend Framework	React (HTML/CSS/JS)
Backend Framework	Python
Database	MS SQL Server, SQL Server Management Studio
IDE	VS Code, GitHub Codespaces
Communication	MS Teams, GitHub Issues

8. SCHEDULE SUMMARY WITH MILESTONES

Task dependencies: Database Setup (2.1) must complete before Backend API (2.2) integration. Backend API must complete before Frontend integration (2.3). Phase I must complete before Phase II development begins.

Tasks	Owner	Start Date	Duration (Days)	End Date
1 Planning		7-Jan-2026	21	27-Jan-2026
1.1 Team Formation	Michael	7-Jan-2026	7	13-Jan-2026
1.2 Project Plan	Michael	7-Jan-2026	14	20-Jan-2026
1.3 Project Design	Team	14-Jan-2026	14	27-Jan-2026
2 Development Phase I		21-Jan-2026	14	3-Feb-2026
2.1 Database Setup	Tony	21-Jan-2026	10	31-Jan-2026
2.2 Backend API	Sean	21-Jan-2026	14	3-Feb-2026
2.3 Basic Frontend	Ishan	24-Jan-2026	11	3-Feb-2026
2.4 Peer Review I	Team	3-Feb-2026	1	3-Feb-2026
3 Testing		28-Jan-2026	28	24-Feb-2026
3.1 Test Plan	Bryana	28-Jan-2026	14	10-Feb-2026
3.2 Unit Testing	Team	4-Feb-2026	7	10-Feb-2026
3.3 Integration Testing	Bryana	11-Feb-2026	14	24-Feb-2026
4 Development Phase II		4-Feb-2026	21	24-Feb-2026
4.1 Dashboard/Analytics	Ishan	4-Feb-2026	14	17-Feb-2026
4.2 Recommendations	Sean	4-Feb-2026	14	17-Feb-2026
4.3 Bug Fixes/Polish	Team	18-Feb-2026	7	24-Feb-2026
5 Documentation		11-Feb-2026	21	3-Mar-2026
5.1 User Guide	Michael	11-Feb-2026	14	24-Feb-2026
5.2 Final Report	Michael	18-Feb-2026	14	3-Mar-2026
5.3 Peer Review II	Team	3-Mar-2026	1	3-Mar-2026

Note: All deadlines are 11:59 PM on the due date.

9. SCHEDULE DETAIL (GANTT CHART)

Summary:

Phase		January			February				March
Detail	Target End	13	20	27	3	10	17	24	3
Meet Team	13-Jan	🟩							
Project Plan	20-Jan	🟩	🟩						
Project Design	27-Jan		🟦	🟦					
Phase 1 Build	3-Feb			🟦	🟦				
Test Plan	10-Feb				🟦	🟦			
Phase 2 Build	17-Feb					🟦	🟦		
User Guide	24-Feb						🟦	🟦	
Final Report	3-Mar							⬛	⬛

Detail View:

Phase			January			February				March
Detail	Target Start	Target End	13	20	27	3	10	17	24	3
Meet Team	7-Jan	13-Jan	Complete							
Team Formation		10-Jan	Complete							
Discuss Roles		12-Jan	Complete							
Communication Plan		13-Jan	Complete							
Project Plan	7-Jan	20-Jan	🟩	Complete						
Draft Sections		15-Jan	🟩	Complete						
Team Review		18-Jan		Complete						
Final Submission		20-Jan		Complete						
Project Design	14-Jan	27-Jan	🟦	🟦	🟦					
Database Schema		24-Jan		🟦	🟦					
API Design		24-Jan		🟦	🟦					
UI Wireframes		24-Jan		🟦	🟦					
Final Submission		27-Jan			🟦					
Phase 1 Build	21-Jan	3-Feb	🟦	🟦	🟦	🟦				
Database Setup		31-Jan		🟦	🟦	🟦				
Backend API		3-Feb		🟦	🟦	🟦				
Basic Frontend		3-Feb		🟦	🟦	🟦				
Peer Review 1		3-Feb	🟦	🟦	🟦	🟦				
Test Plan	28-Jan	10-Feb		🟦	🟦	🟦	🟦			
Test Cases		7-Feb		🟦	🟦	🟦	🟦			
Test Environment		7-Feb		🟦	🟦	🟦	🟦			
Final Submission		10-Feb					🟦			
Phase 2 Build	4-Feb	17-Feb				🟦	🟦	🟦		
Dashboard/Charts		14-Feb				🟦	🟦	🟦		
Recommendations		14-Feb				🟦	🟦	🟦		
Integration		17-Feb						🟦		
User Guide	11-Feb	24-Feb		🟦	🟦	🟦	🟦	🟦	🟦	
Draft Guide		21-Feb		🟦	🟦	🟦	🟦	🟦	🟦	
Screenshots		21-Feb		🟦	🟦	🟦	🟦	🟦	🟦	
Final Submission		24-Feb							🟦	
Final Report	18-Feb	3-Mar				🟦	🟦	🟦	🟦	🟦
Compile Report		1-Mar					🟦	🟦	🟦	🟦
Peer Review 2		3-Mar				🟦	🟦	🟦	🟦	🟦
Presentation		3-Mar								🟦

10. COMMUNICATION PLAN

10.1 Contact List

Name	Organization	Role	Emergency Communication
Michael	UMGC CMSC495	Project Manager	Michael.allen.us217@gmail.com
Ishan	UMGC CMSC495	Frontend Lead	ishan.akhouri@gmail.com
Sean	UMGC CMSC495	Backend Lead	skellner2@student.umgc.edu
Bryana	UMGC CMSC495	Test/ QA Lead	bhenderson56@student.umgc.edu
Tony	UMGC CMSC495	Database Lead	tarista@student.umgc.edu
Lynda Metallo	UMGC CMSC495	Instructor	lynda.metallo@faculty.umgc.edu

10.2 Communication Tools

Tool	Purpose
MS Teams	Primary communication: daily chat, quick questions, syncs
GitHub	Code repository, issue tracking, pull requests
MS Word (Teams)	Collaborative document drafting

10.3 Meeting Schedule

Saturday Sync: 3:30 PM EST - Full team (mandatory) **Async Check-ins:** Monday and Thursday via MS Teams - post status updates, flag blockers **Daily:** Quick updates in Teams chat as needed

10.4 Decision Log

Major decisions (scope changes, tech pivots) documented in Teams or GitHub Issues with date, decision, and rationale.

10.5 Team Norms

- Respond to messages within 24 hours
- Attend scheduled syncs or notify in advance
- Push code to your branch at least every 2 days
- Flag blockers early—no surprises
- Treat each other with dignity and respect

11. RISK MANAGEMENT

During development, several risk factors can result in project delays or negatively affect the project. The following risks have been identified along with mitigation strategies.

Risk	Prob	Impact	Mitigation Strategy
Team Availability / Time Zones	High	High	Weekly Saturday syncs accommodate all. Async communication via Teams. Recorded meetings.
Team Member Drops Course	Low	High	Work on all components as a team. Document code thoroughly. No single point of failure.
Unfamiliarity with Tech Stack	Med	Med	Select technologies team already knows. (Python, React, HTML/CSS/JS, MS SQL Server).
Scope Creep	Med	High	Firm scope boundaries in Section 3. PM gatekeeps new features.
Integration Issues	Med	Med	Define API contracts early. Integrate continuously. Use feature branches.
Schedule Conflict Issues	High	Low	Will complete action items before absence, monitor from phone, engage and ask for help early

12. EVALUATION CRITERIA

We will measure success by:

- All eight functional requirements working in final demo
- Test Plan achieving >80% test case pass rate
- User Guide reviewed and approved by a non-team member
- Team sends deliverables on time
- Peer review averages ≥7/10 across all categories

TEAM APPROVAL

By submitting this document, all team members confirm agreement with this project plan.

Team Member	Role	Date
Ishan	Frontend Lead	1/19/2026
Michael	Project Manager	1/20/2026
Bryana	Test/ QA Lead	1/20/2026
Sean	Backend Lead	1/20/2026
Tony	Database Developer	1/18/2026