User Guide

Cloud Cost Intelligence Platform

Michael, Ishan, Sean, Bryana, Tony

CMSC 495 Computer Science Capstone

Instructor: Lynda Metallo

February 24, 2026

Table of Contents

1. Introduction

1.1 Purpose

This User Guide provides step-by-step instructions for setting up and using the Cloud Cost Intelligence Platform, a web-based, multi-cloud cost-monitoring dashboard developed as a capstone project for CMSC 495 at the University of Maryland Global Campus. It is intended for cloud administrators, FinOps practitioners, DevOps engineers, and IT decision-makers who need visibility into aggregated cloud spending, waste-detection insights, and cost-optimization recommendations across AWS, Azure, and GCP environments.

1.2 Overview of the Software

The Cloud Cost Intelligence Platform is a three-tier web application that aggregates simulated cloud cost data from AWS, Azure, and GCP into a unified dashboard. The platform enables organizations to visualize spending trends over time, identify underutilized or idle resources through automated waste detection algorithms, receive rightsizing recommendations to reduce costs, configure per-client budget thresholds with alert notifications, and export formatted reports in CSV and PDF formats. The application is built with an HTML/CSS/JavaScript frontend using Chart.js for data visualization, and a Python Flask backend that exposes approximately 40,000 simulated usage records. All services are containerized using Docker and served through an nginx reverse proxy.

1.3 Scope

This guide covers all implemented functional requirements of the Cloud Cost Intelligence Platform v1.0, including cost dashboard views, trend analysis, provider filtering, waste detection, rightsizing recommendations, budget thresholds, export functionality, and usage metrics. It applies to the web-based application accessed via modern browsers (Chrome, Firefox, Edge) within the project's Docker-based demonstration environment and reflects the scope boundaries defined in the Project Plan. Features documented as future enhancements are noted but not covered in procedural detail.

1.4 Intended Audience

This guide is intended for the following users:

Cloud Administrators — Responsible for managing cloud infrastructure and monitoring costs across AWS, Azure, and GCP environments.

FinOps Practitioners — Financial operations professionals who analyze spending and recommendations

IT Managers — Decision-makers who need summary-level cost visibility and budget compliance reporting

DevOps Engineers - Technical users who analyze resource utilization, investigate waste alerts, and implement optimization recommendations.

1.5 Document Conventions

Convention	Meaning	Example
Bold Text	UI elements: buttons, menus, labels, page names	Click **Save Settings**
`Monospace`	Commands, code, file paths, URLs	`Docker-compose up --build`
Step 1, Steps 2…	Sequential user actions — follow in order	Step 1: Navigate to http://localhost
Menu → Page	Navigation path through the interface	**Settings → Export Reports**
⚠ Warning	Important caution or potential issue	⚠ change this to 0.0.0.0 for Docker
💡 Tip	Helpful suggestion, not required	💡 CSV files are faster to generate

2. System Requirements

2.1 Hardware Requirements

Component	Minimum	Recommended
Processor	Dual-core 2.0 GHz	Quad-core 2.5 GHz+
RAM	4 GB	8 GB+
Display	1280 x 720	1920 x 1080+
Network	Broadband connection	10 Mbps+

2.2 Software Requirements

Software	Requirement
Operating System	Windows 10+, macOS 12+, Ubuntu 22.04+
Web Browser	Chrome 90+, Firefox 90+, Edge 90+, Safari 15+
JavaScript	Enabled in browser settings

2.3 Developer / Local Setup Requirements

For developers running the application locally or for instructor evaluation:

Component	Version	Notes
Docker Desktop	4.0+	Required for containerized deployment
Git	2.30+	Repository cloning
Python	3.11+	Backend (Flask)
Node.js	18+	Frontend testing (Jest)
Azure SQL Access	N/A	UMGC student or faculty Microsoft online umgc.edu account
ODBC Driver 18	18+	Required for pyodbc database connectivity
Azure CLI	2.50+	Required if using device code authentication

3. Installation and Setup

3.1 Accessing the Application

The Cloud Cost Intelligence Platform runs locally via Docker containers. There is no hosted production deployment for this capstone release. Users access the application by completing the Local Development Setup below and navigating to http://localhost:8080 in their browser.

3.2 Local Development Setup

Step 1: Clone the Repository

- `git clone https://github.com/michaelallenus217-lang/CMSC495-CloudCost`
- `cd CMSC495-CloudCost`

Step 2: Configure Environment Variables
From the project root directory (`CMSC495-CloudCost/`), copy the provided template:

- `cp src/backend/.env.example src/backend/.env`
- Open `src/backend/.env` in any text editor (VS Code, Notepad, nano, or vim) and set the following values:

```
ENV=local
AZURE_IDENTITY_MODE=devicecode
AZURE_SQL_SERVER=cmsc495-cloud-cost.database.windows.net
AZURE_SQL_DATABASE=CloudCostDatabase
FLASK_HOST=0.0.0.0
FLASK_PORT=5000
CORS_ORIGINS=http://localhost:8080
```

⚠ **Warning:** The .env.example ships with FLASK_HOST=127.0.0.1. You must change this to 0.0.0.0 for Docker. Without this change, the backend will not be reachable from the frontend container.

⚠ **Warning:** Never commit the .env file to version control. The .gitignore excludes it by default.

Step 3: Verify access to Azure SQL database using Microsoft Entrata ID (first-time use)
Verify the user account can access database resources using the @umgc.edu account.

1. Open a web browser, go to https://portal.azure.com
   From **Pick an Account** prompt, sign in with your @umgc.edu account.

2. In the search bar at the top, type “Azure SQL Database” and select it under **Services**

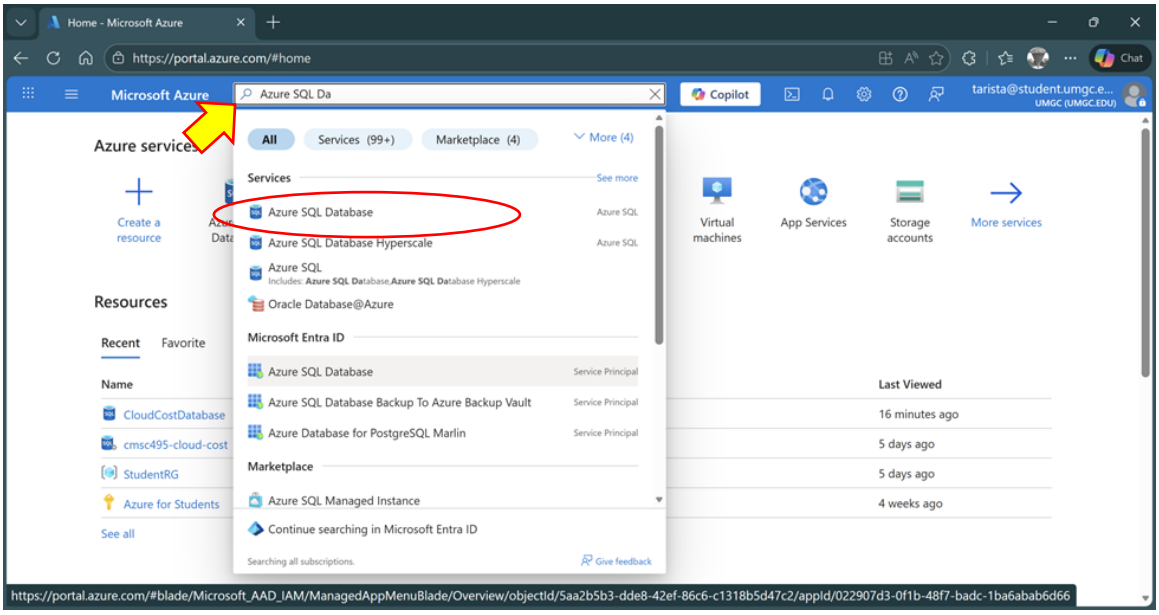

Figure 1: Searching for database service in Microsoft Azure

3. Select the database “CloudCostDatabase”

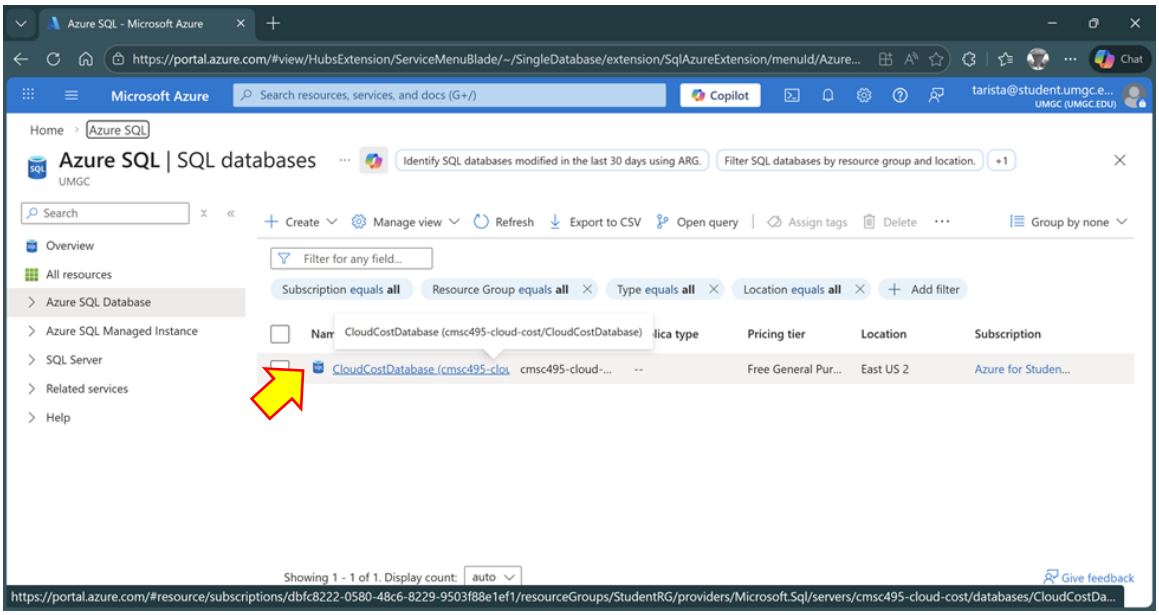

Figure 2: Selecting the CloudCostDatabase

4. In the left menu, select **Query Editor (preview)**. When prompted for authentication, select **Microsoft Entra** and continue using your @umgc.edu account.

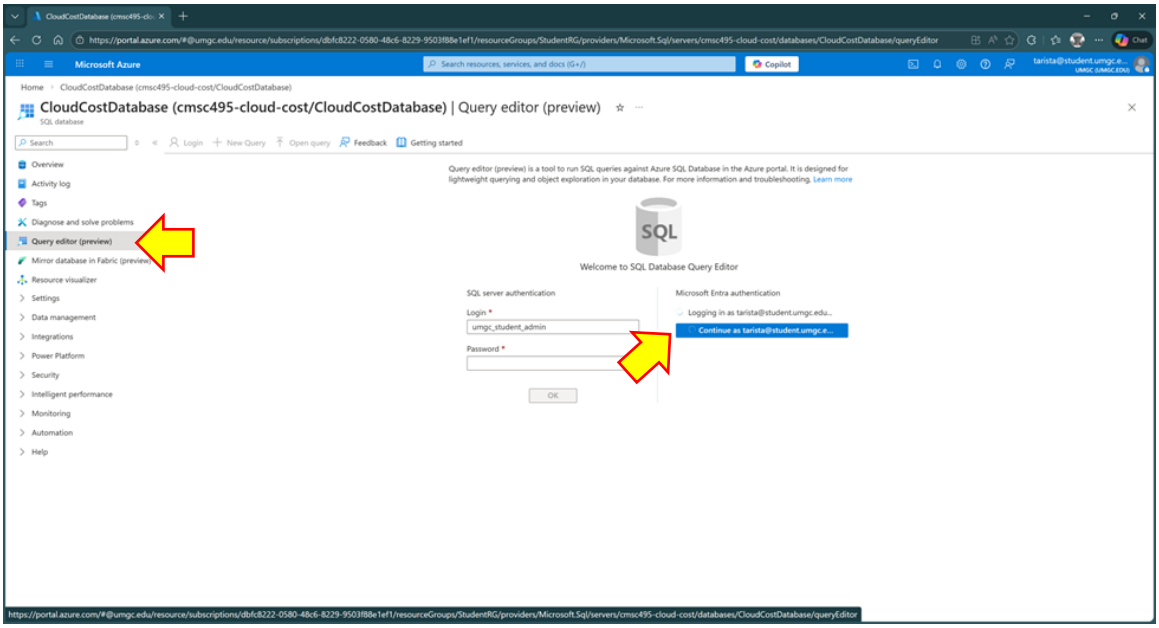

Figure 3: Selecting the Query editor and authenticating access

Note: On the first connection or when the system IP address changes, the database firewall will require a rule to grant access. If you receive the following error: Cannot open server 'cmsc495-cloud-cost' requested by the login. Client with IP address “your ip address” is not allowed to access the server.

Click the link: Allowlist IP "your ip address” on server cmsc495-cloud-cost. A pop-up will appear stating: “Successfully update server firewall rules”

5. A successful connection will show the Query editor and the contents of the database. Once access has been confirmed, close your browser.

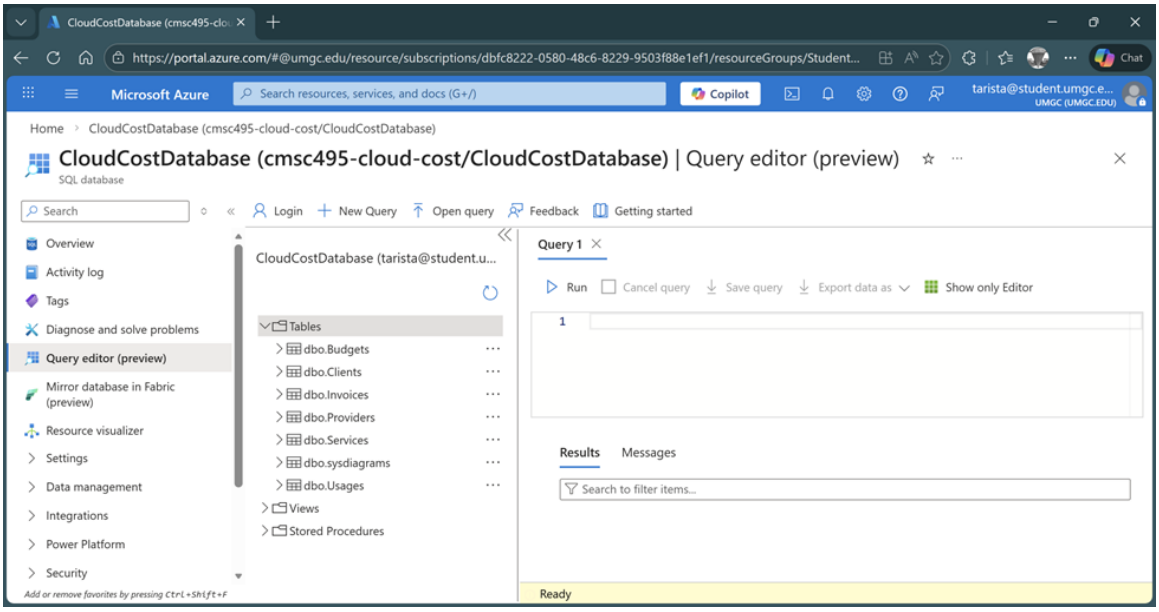

Figure 4: Viewing the content of the database in Query editor

Step 4: Build and Start Docker Containers

Execute the Docker build command on the command line:

```
docker compose up --build
```

This command builds and starts two services: the Flask backend API and the nginx frontend server. The initial build may take several minutes as Docker downloads base images and installs dependencies.

⚠️ **Warning (Apple Silicon users):** If running on an ARM-based Mac (M1/M2/M3/M4), prefix the command with: DOCKER_DEFAULT_PLATFORM=linux/amd64 docker compose up --build

Step 5: Authenticate with Azure:

The backend will prompt for device code authentication twice (this is expected in the development environment due to Flask's debug reloader). Complete both prompts:

1. Watch the terminal for a message: "To sign in, use a web browser to open the page https://login.microsoft.com/device and enter the code XXXXXXXXX"
2. Open the URL in your browser
3. Enter the code displayed in the terminal
4. Sign in with your UMGC Microsoft account (@umgc.edu)
5. A second prompt will appear with a new code — repeat steps 2–4 using https://microsoft.com/devicelogin
6. After both authentications have finished, the terminal will display: "Debugger is active!"

💡 **Tip:** URLs: login.microsoft.com/device or microsoft.com/devicelogin work for either prompt.

Step 6: Verify the Application
Open a browser and navigate to http://localhost:8080. The dashboard should load with cost data.

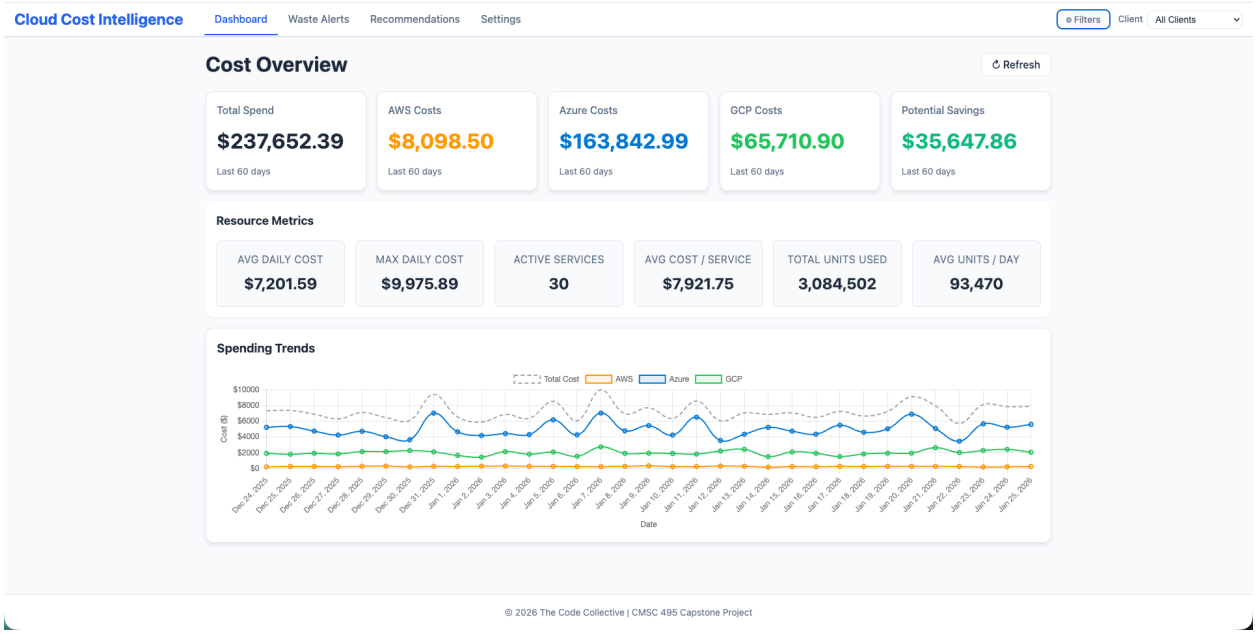

Figure 5: Dashboard Overview of the Cloud Cost Intelligence Platform.

NOTE: The first data request may take 30–60 seconds due to Azure SQL Serverless cold start. Subsequent requests respond normally.

Step 7: Stopping the Application:

Press Ctrl+C in the terminal running Docker, or run:

```
docker compose down
```

This stops and removes all containers. Your data is preserved in the Azure SQL database, and your .env file remains on disk.

4. Getting Started

4.1 Logging In

Authentication is handled via device code flow during Docker startup (see Section 3.2, Step 4). There is no browser-based login page. Once both device code prompts are completed in the terminal, open a browser and navigate to http://localhost:8080. The dashboard loads automatically with your authenticated session.

💡 **Tip:** If the dashboard shows no data, verify that both device code authentications completed successfully and review the Azure SQL database access in Section 3.2, Step 3.

4.2 Dashboard Overview

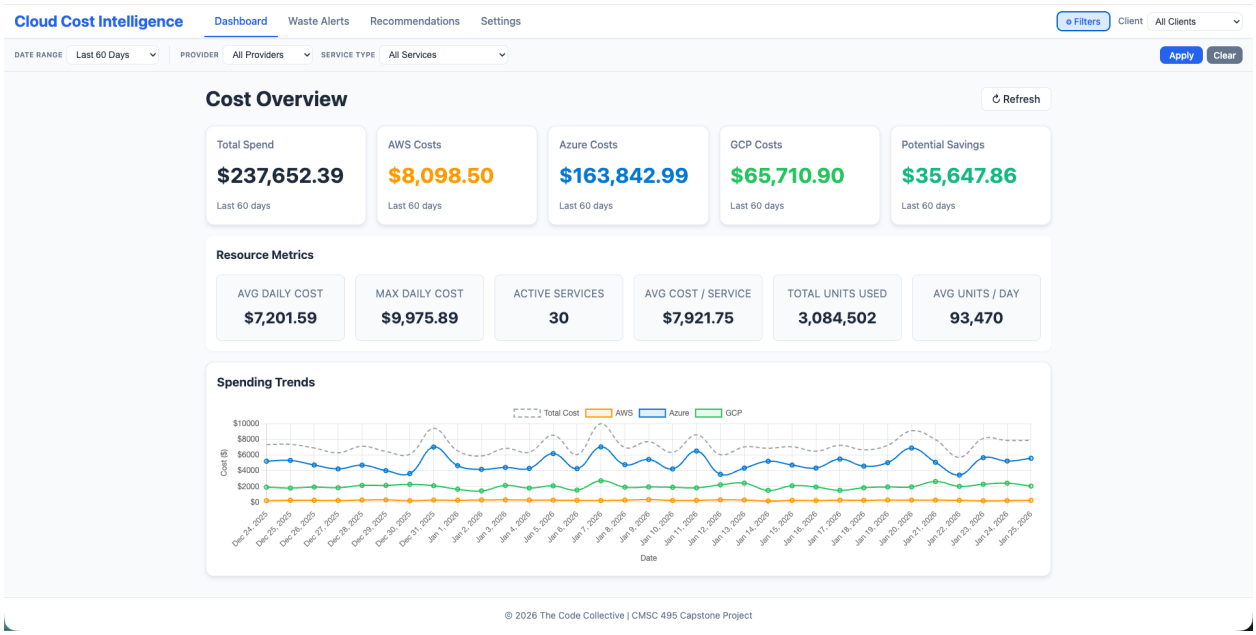

Figure 6: Main Dashboard Overview

Area	Description
Navigation Bar	Top bar with **Dashboard**, **Waste Alerts**, **Recommendations**, and **Settings** tabs. Includes **Filters** toggle button and **Client** dropdown on the right.
Cost Overview Cards	Five summary cards: Total Spend, AWS Costs, Azure Costs, GCP Costs, and Potential Savings. Each shows the dollar amount for the selected date range.
Filter Controls	Hidden by default. Click the **Filters** button to reveal Date Range, Provider, and Service Type dropdowns. Click **Apply** to filter or **Clear** to reset.
Resource Metrics	Six metric cards: Avg Daily Cost, Max Daily Cost, Active Services, Avg Cost/Service, Total Units Used, and Avg Units/Day.
Spending Trends	Chart.js line chart showing cost trends over time with dates on the X-axis and cost in USD on the Y-axis.
Footer	Copyright notice: "© 2026 The Code Collective"

4.3 Navigating the Interface

The application uses a top navigation bar with four main views:

- **Dashboard -** Default landing page. Displays cost overview cards, resource metrics, and spending trends chart. Click **Filters** to reveal date range, provider, and service type controls. Use the **Client** dropdown in the top right to switch between client datasets. Click **Refresh** to reload data from the server.

- **Waste Alerts -** Displays a table of underutilized resources flagged by the waste detection engine. Sortable by column headers.

- **Recommendations -** Shows rightsizing recommendation cards generated from the top waste alerts. Each card includes the current configuration, the suggested alternative, and the estimated savings.

- **Settings - Budget configuration** to set per-client budget thresholds, alert percentages, and **report export** to generate cost data reports in CSV or PDF format.

Note: Navigation is persistent; the top bar remains visible on all pages. The currently active page is highlighted in the nav bar.

5. Features and How-To Guides

5.1 Viewing Cost Summary

The dashboard cost summary provides an at-a-glance view of total cloud spending across all providers. The summary cards display five key metrics:

Card	Description
Total Spend	Aggregated cost across all providers for the selected date range
AWS Costs	Amazon Web Services spend for the selected period
Azure Costs	Microsoft Azure spend for the selected period
GCP Costs	Google Cloud Platform spend for the selected period
Potential Savings	Estimated monthly savings from optimization opportunities

Data is aggregated from raw usage records in the database, grouping costs by provider and summing across the selected date range.

Below the cost overview cards, the **Resource Metrics** row displays six operational indicators:

Metric	Description
Avg Daily Cost	Average daily spend across the selected date range
Max Daily Cost	Highest single-day cost recorded in the selected period
Active Services	Count of distinct cloud services with usage data
Avg Cost/Service	Average cost per active service
Total Units Used	Sum of all resource units consumed
Avg Units/Day	Average daily resource consumption

To view the cost summary:

- Navigate to the **Dashboard** view (default landing page)
- Select a date range from the dropdown (Last 7 Days, Last 30 Days, Last 60 Days, or Last 90 Days)
- Click the **Refresh** button to reload data
- View the five summary cards at the top of the dashboard
- Review the Resource Metrics row below the summary cards for operational indicators
- The "Last N days" label updates automatically to reflect your selected time period

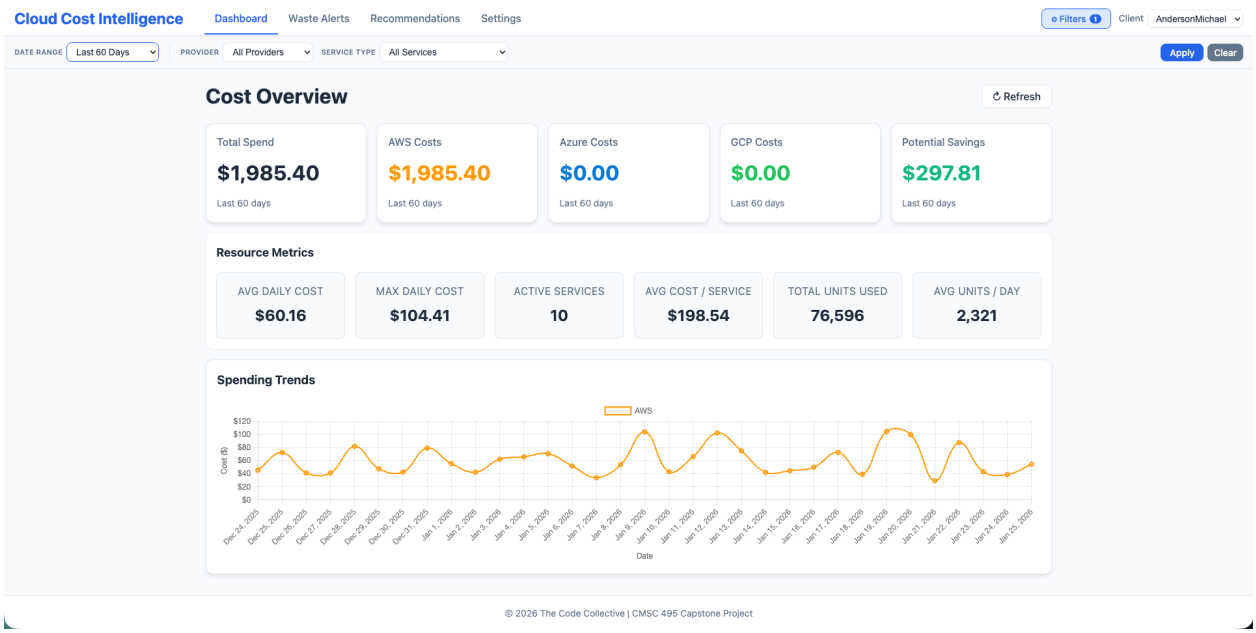

Figure 7: Dashboard showing Cost Overview cards, Resource Metrics, and Spending Trends

5.2 Filtering and Sorting Data

The filter controls allow users to narrow dashboard data by date range, cloud provider, service type, and client. Filters are hidden by default to maximize dashboard viewing area. When filters are applied, all dashboard components (summary cards, resource metrics, trend charts, waste alerts) update to reflect the filtered dataset.

Filter	Options	Description
Date Range	Last 7 Days, Last 30 Days, Last 60 Days, Last 90 Days	Controls the time window for all displayed data
Provider	All Providers, AWS, Azure, GCP	Narrows data to a specific cloud platform
Service Type	All Services, *(varies by provider)*	Filters by cloud service category
Client	All Clients, *(client names)*	Filters all data to a specific client account

To apply filters:

- Click the **Filters** button in the top navigation bar to reveal the filter controls
- Select options from the **Date Range**, **Provider**, and/or **Service Type** dropdowns
- To filter by client, use the **Client** dropdown in the top-right corner of the navigation bar
- Click **Apply** to update the dashboard
- Click **Clear** to reset all filters and return to the full dataset

Filtering is performed client-side on cached data, providing instant results without triggering additional API calls.

💡 **Tip:** The Client dropdown persists across all pages (Dashboard, Waste Alerts, Recommendations, Settings). Select a client first, then navigate between views to see that client's data throughout.

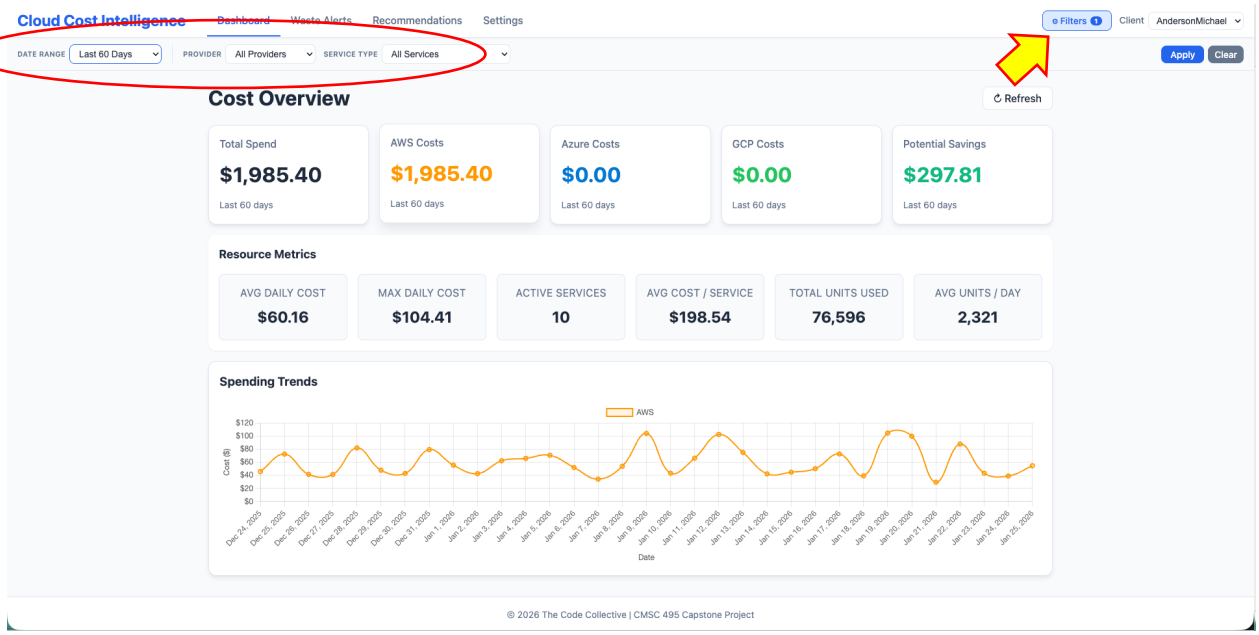

Figure 8: Filter controls revealed after clicking the Filters button

5.3 Viewing Spending Trends

The spending trends chart visualizes cost patterns over time using Chart.js line visualization. The chart displays four data series:

Line	Color	Description
Total Cost	Grey *(dash)*	Combined spend across all providers
AWS	Orange	Amazon Web Services spend
Azure	Blue	Microsoft Azure spend
GCP	Green	Google Cloud Platform spend

The X-axis represents dates within the selected range, while the Y-axis shows cost in USD. Hovering over any data point reveals a tooltip with the exact date and cost value for that day.

Chart features:

Feature	Description
Time Axis	Displays dates in chronological order from oldest to newest
Cost Axis	Automatically scales based on maximum cost in the dataset
Legend	Color-coded labels identify each provider and total cost line
Interactive tooltips	Hover over data points to see precise values
Responsive design	Chart resizes automatically based on browser window size

To interpret the trend chart:

- Upward slopes indicate increasing spending over time
- Flat sections suggest consistent daily costs
- Gaps in data lines indicate days with no recorded usage
- Compare provider lines to identify which provider drives total costs

💡 **Tip:** Use the **Date Range** filter to zoom into shorter periods for more granular trend analysis, or expand to 90 days for long-term patterns.

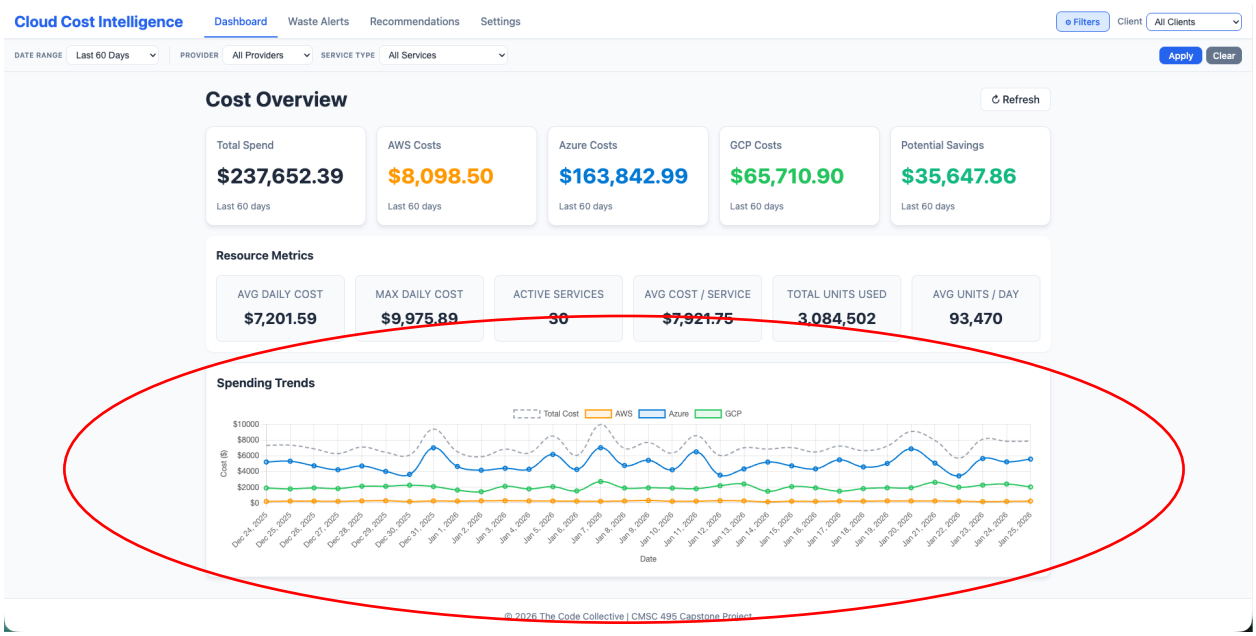

Figure 9: Spending Trends chart showing cost patterns across providers over time

5.4 Budget Tracking

The budget tracking feature allows users to set monthly spending limits and alert thresholds for specific clients. Budget settings are configured on the **Settings** page and are saved per client, persisting across sessions.

To configure budget settings:

- Select a client from the **Client** dropdown in the top-right navigation bar
- Navigate to the **Settings** page via the top navigation menu
- Enter the monthly budget amount in the **Monthly Budget ($)** field (e.g., 1500)
- Enter the monthly limit in the **Monthly Limit ($)** field (typically higher than budget, e.g., 1800)
- Adjust the **Alert Threshold (%)** slider to set the alert trigger point (default 50%)
- Check or uncheck **Enable Budget Alerts** to activate or deactivate alerts
- Click **Save Settings** to persist changes
- Click **Cancel** to discard changes without saving

Field	Description	Example
Monthly Budget ($)	Target spending amount for the selected client	1500
Monthly Limit ($)	Hard spending cap — should be higher than budget	1800
Alert Threshold (%)	Percentage of budget that triggers an alert	50%
Enable Budget Alerts	Activates or deactivates alert notifications	Checkbox

Budget alert logic:

- Alerts trigger when actual spending reaches the threshold percentage of the budget amount
- Example: $1,500 budget with 50% threshold = alert at $750 spent
- Alerts are computed dynamically by comparing current month spending to budget configuration

⚠️ **Warning:** You must select a client from the **Client** dropdown in the top navigation bar before configuring budget settings. Settings apply to the currently selected client only.

💡 **Tip:** The budget target you set here is displayed on the **Waste Alerts** and **Recommendations** pages, where it is used to calculate whether spending is on track or over budget.

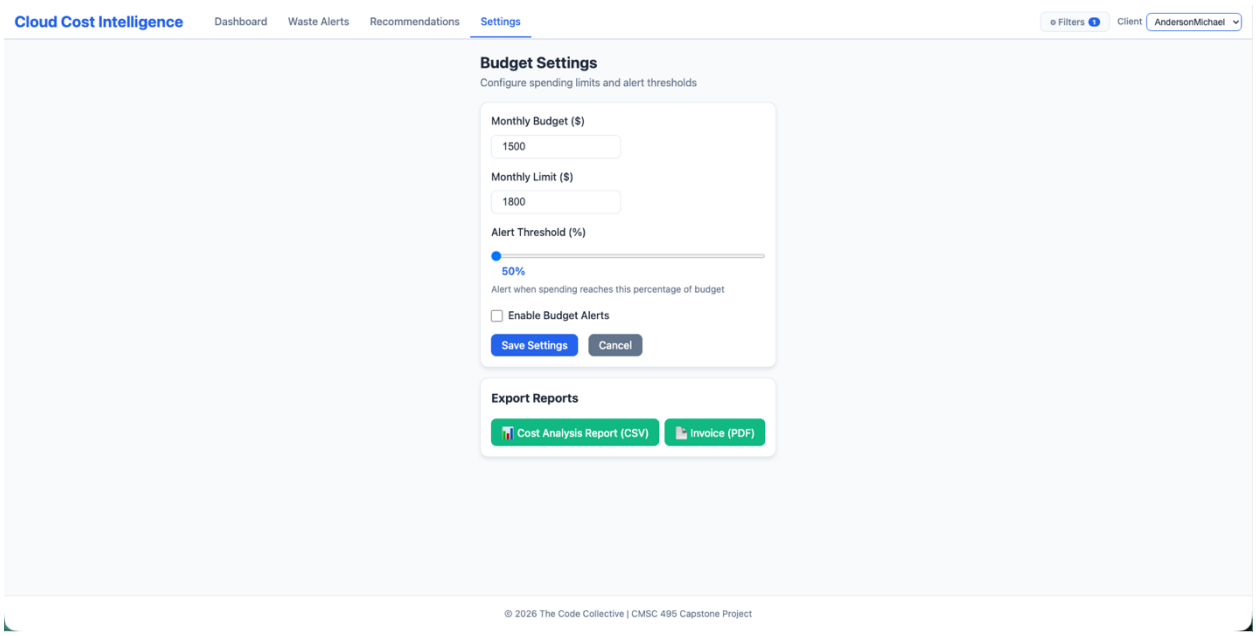

Figure 10: Settings page — Budget configuration and export options

5.5 Waste Alerts and Cost Anomaly Detection

The Waste Alerts page identifies cloud resources with low utilization or optimization opportunities. It provides a comprehensive view of current spending health, severity-ranked resource analysis, and category-level cost breakdowns.

Summary Cards

The top of the page displays four spending health indicators:

Card	Description
Monthly Spend	Total cost for the selected client and date range
Recoverable Waste	Estimated monthly savings available from optimization
Budget Target	Budget amount configured in Settings for the selected client
Over Budget	Status indicator — **On Track** (green) if under budget, warning/alert if over

Severity Indicators
Below the summary cards, three counters categorize all monitored resources:

Severity	Indicator	Meaning
Critical	Red	Severely underutilized — immediate action recommended
Warning	Yellow	Below optimal utilization — review recommended
Optimized	Green	Utilization within acceptable range

NOTE: A yellow diversification banner may appear if all spending is concentrated with a single provider (e.g., "100% of spend is with Amazon Web Service. No cross-provider alternatives available for diversification."). This indicates a vendor lock-in risk.

Waste Alerts Table
The main table lists all monitored resources with the following columns:

Column	Description
Severity	Color-coded indicator (Critical, Warning, or Info/Optimized)
Service	Resource name and service category (e.g., DynamoDB — Databases)
Provider	Cloud provider (Amazon Web Service, Azure, GCP)
Utilization	Current usage percentage — color-coded by threshold
Monthly Cost	Current monthly spend for the resource
Potential Savings	Estimated monthly savings if optimized
Trend	Utilization trend with percentage and direction (▲ up / ▼ down)
Alt Provider	Whether a cross-provider alternative is available (✓ Available) or locked to current provider (— Locked)

To view and analyze waste alerts:

- Navigate to **Waste Alerts** via the top navigation menu
- Review the summary cards for overall spending health
- Check severity counters to understand the distribution of resource health
- Scroll through the table to review individual resources
- Click column headers to sort by Severity, Utilization, Monthly Cost, Potential Savings, or Trend
- Note the **Alt Provider** column to identify opportunities for cross-provider migration

Interpreting waste alerts:

- **Warning severity with low utilization (< 50%):** Strong candidate for rightsizing or configuration changes
- **Info/Optimized with moderate utilization (50–70%):** Monitor for trends — rising utilization is positive, declining may indicate emerging waste
- **High Trend percentage (▲):** Utilization is increasing — resource may be right-sizing itself through organic growth
- **Negative Trend (▼):** Utilization is declining — investigate for potential waste

💡 **Tip:** Select a specific client from the **Client** dropdown before reviewing waste alerts. Each client's resource portfolio has different utilization patterns.

Spend by Category
Below the alerts table, a horizontal bar chart breaks down monthly costs by service category:

Category	Description
File Storage	EFS, Azure Files, and equivalent services
Object Storage	S3, Azure Blob, GCP Cloud Storage
Compute	EC2, Lambda, Batch, and equivalent services
Databases	RDS, DynamoDB, and equivalent services
Containers	EKS, Fargate, and equivalent services
Application Deployment	Elastic Beanstalk, App Service, and equivalent services

This chart helps identify which service categories drive the most cost and where optimization efforts should focus.

Tip: Cross-reference the Spend by Category chart with the alerts table. If a category has high spend but all resources show Optimized status, the spending may be justified by high utilization.

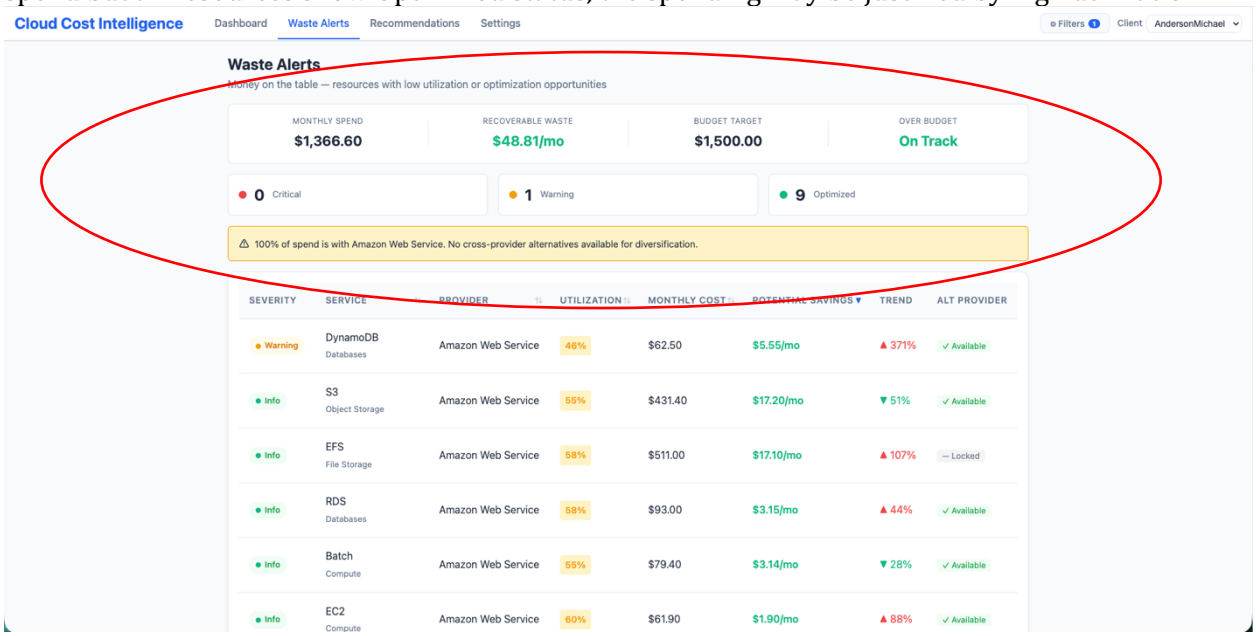

Figure 11: Waste Alerts overview showing spending health, severity counters, and diversification warning

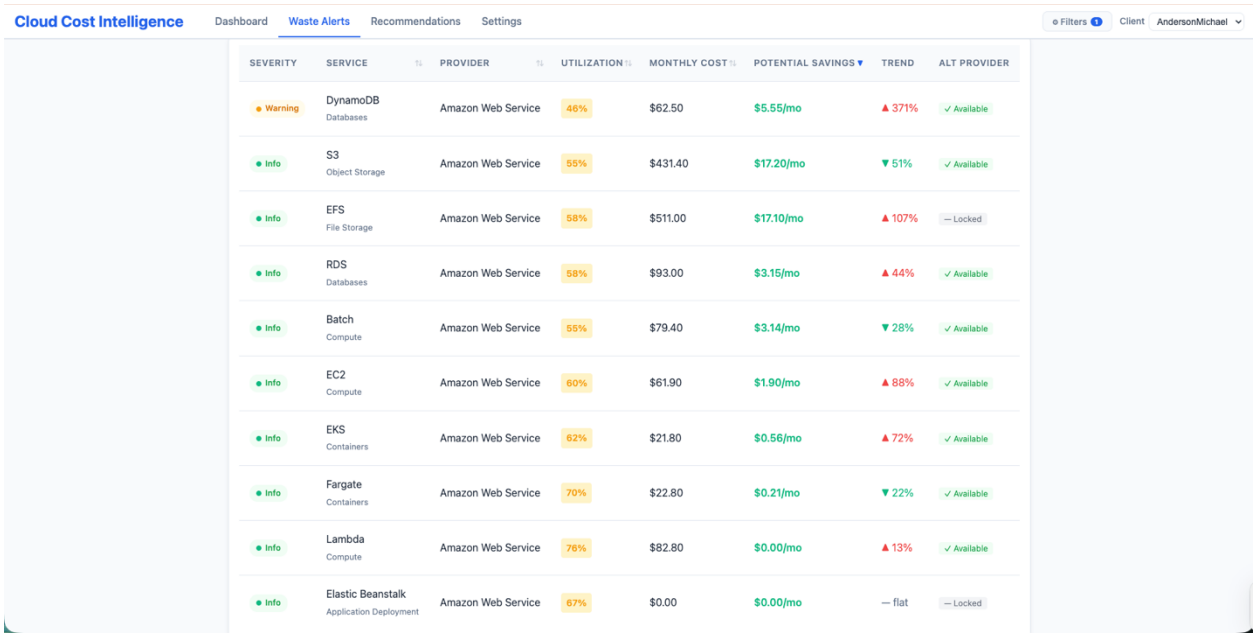

Figure 12: Waste Alerts table with severity, utilization, savings, trend, and alternative provider columns

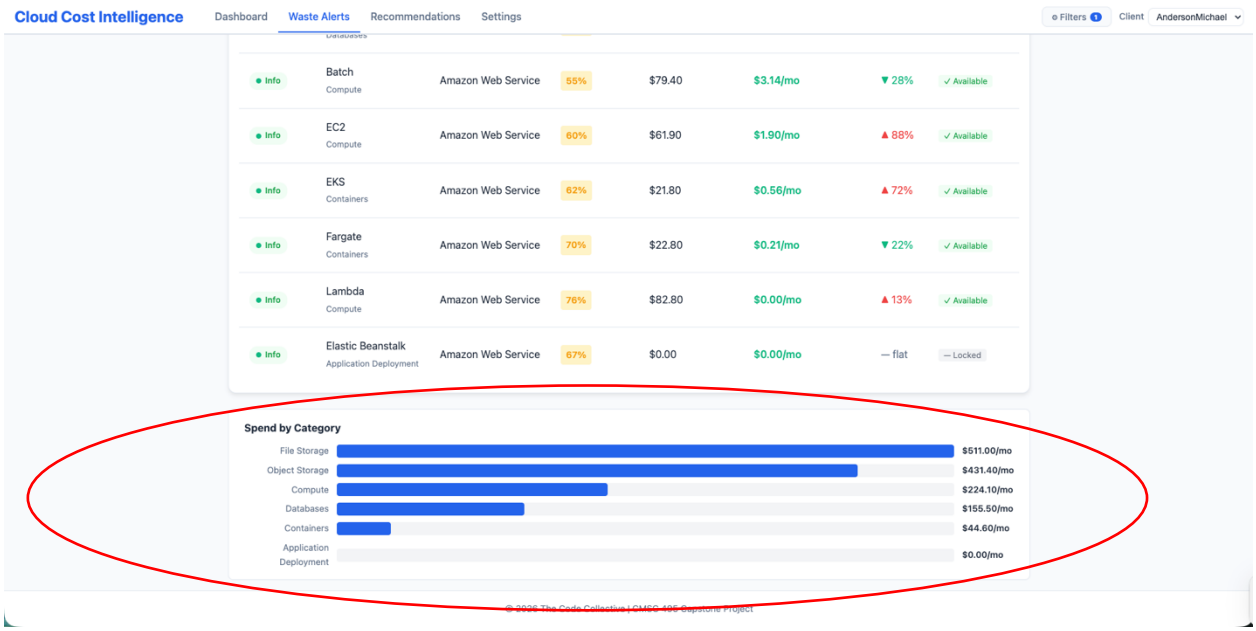

Figure 13: Spend by Category horizontal bar chart

5.6 Rightsizing Recommendations

The Optimization Recommendations page presents a phased savings roadmap that analyzes current spending against the client's budget target and generates actionable optimization strategies organized by implementation timeline and complexity.

Savings Summary
The top of the page displays four financial indicators:

Card	Description
Current Spend	Actual monthly spend for the selected client
Budget	Budget target configured in Settings
Optimized Spend	Projected monthly spend after all recommended optimizations
Total Recoverable	Total potential monthly savings across all phases

Savings Gauge
Below the summary cards, a color-coded horizontal gauge visualizes the relationship between optimized spend, budget target, and current spend on a single scale. Key callouts include:

- Percentage of current spend that is recoverable (e.g., "68% of current spend is recoverable — $1,232.00/mo")
- Whether optimization would bring spending under budget (e.g., "✓ Optimization lands $925.00/mo under budget")

Optimization Phases
Recommendations are organized into three implementation phases, each with increasing complexity and timeline:

Phase	Timeline	Description
Phase 1: Config Changes	Now	Low-effort configuration adjustments with immediate payback. Zero downtime, zero upfront cost.
Phase 2: Right-Sizing	30 Days	Resource resizing that may require upfront investment and brief downtime. Higher savings, longer payback period.
Phase 3: Multi-Cloud Evaluation	90 Days	Cross-provider migration analysis. Only recommended if Phases 1 and 2 do not achieve budget targets.

Each phase card displays:

Field	Description
Monthly Savings	Estimated recurring cost reduction
Upfront Cost	One-time investment required to implement
Effort	Estimated labor hours for implementation
Downtime	Expected service interruption (if any)
Payback	Time to recover upfront costs from savings

💡 **Tip:** Phase 3 (Multi-Cloud Evaluation) may show "✓ Not needed" with the message "Already under budget after Phases 1 & 2" if the first two phases achieve sufficient savings. This is a positive indicator — it means fewer changes are required.

Phase Details
Each phase is expandable. Click the arrow (▸) to reveal individual service-level recommendations:

- Service name and provider (e.g., "S3 (Amazon Web Service)")
- Specific optimization action (e.g., "Storage lifecycle + Intelligent-Tiering")
- Estimated monthly savings for that service

Phase Comparison Table
At the bottom of the page, a comparison table provides a side-by-side view across all three phases:

Metric (Anderson)	Phase 1: Now	Phase 2: 30 Days	Phase 3: 90 Days
Monthly Savings	$781.00/mo	$451.00/mo	Not Needed: Client is underbudget after Phase 1 and Phase 2 recommendations.
Upfront Cost	$0	$1,800.00	
Labor Hours	2.25 hrs	22.5 hrs	
Downtime	None	5 min	
Payback	Immediate	~4 months	
Spend After	$1,026.00/mo	$575.00/mo	

A highlighted tip at the bottom of the page recommends the best starting point based on cost-benefit analysis (e.g., "💡 Phase 1 alone saves $781.00/mo with zero downtime and zero upfront cost. Start there.").

To use the Optimization Recommendations:

- Select a client from the **Client** dropdown in the top navigation bar
- Navigate to **Recommendations** via the top navigation menu
- Review the savings summary to understand current spend vs. budget vs. optimized spend
- Read the savings gauge for a visual snapshot of recoverable spend
- Review each phase card to compare effort, downtime, and payback
- Expand phases to see individual service recommendations
- Use the Phase Comparison table to decide which phases to implement
- Start with Phase 1 for immediate, low-risk savings

NOTE: Recommendations are advisory only and do not automatically modify cloud resources. Users should validate suggestions against business requirements and conduct internal change management reviews before implementation.

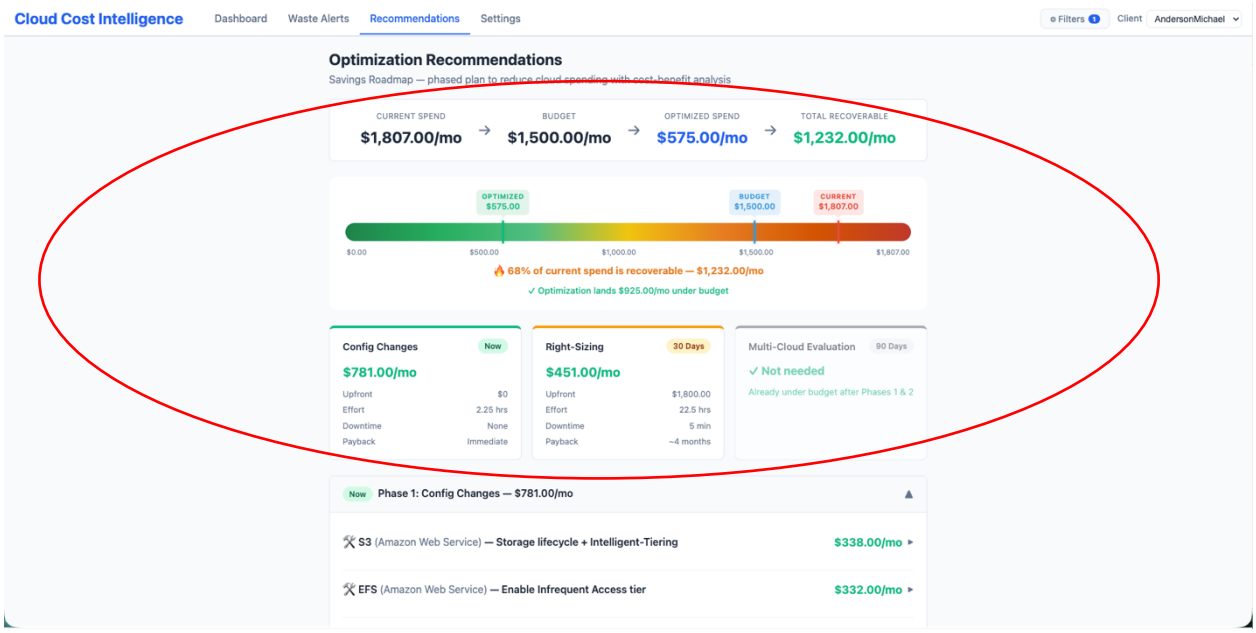

Figure 14: Optimization Recommendations showing savings roadmap with three implementation phases

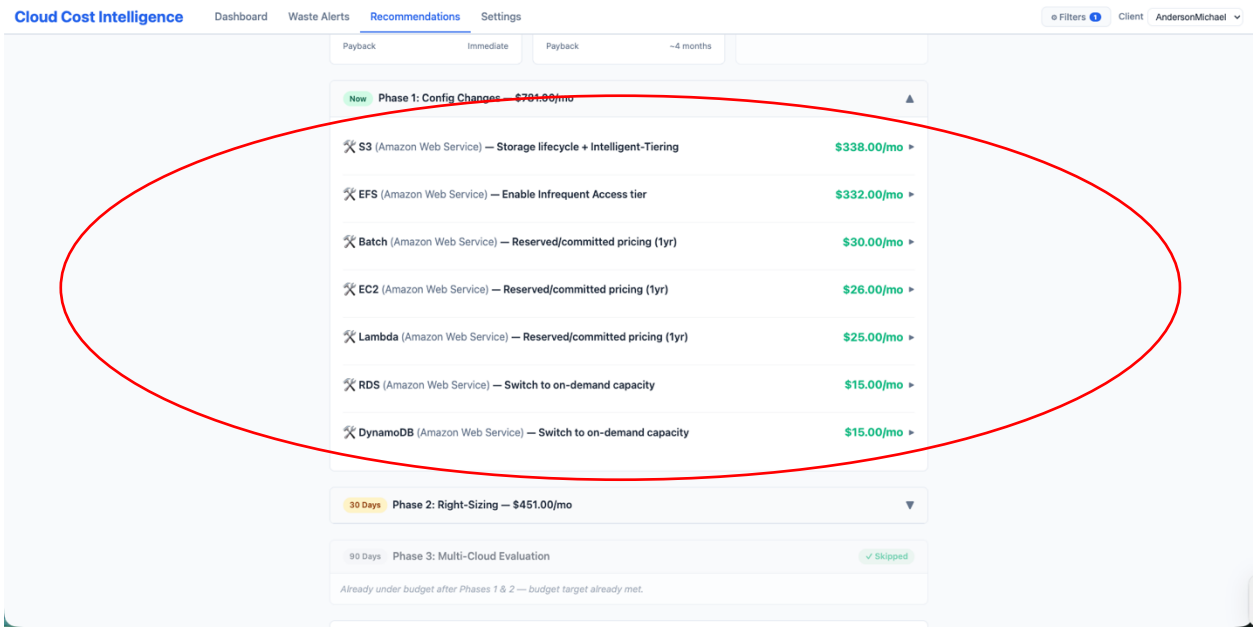

Figure 15: Phase 1 service-level recommendations

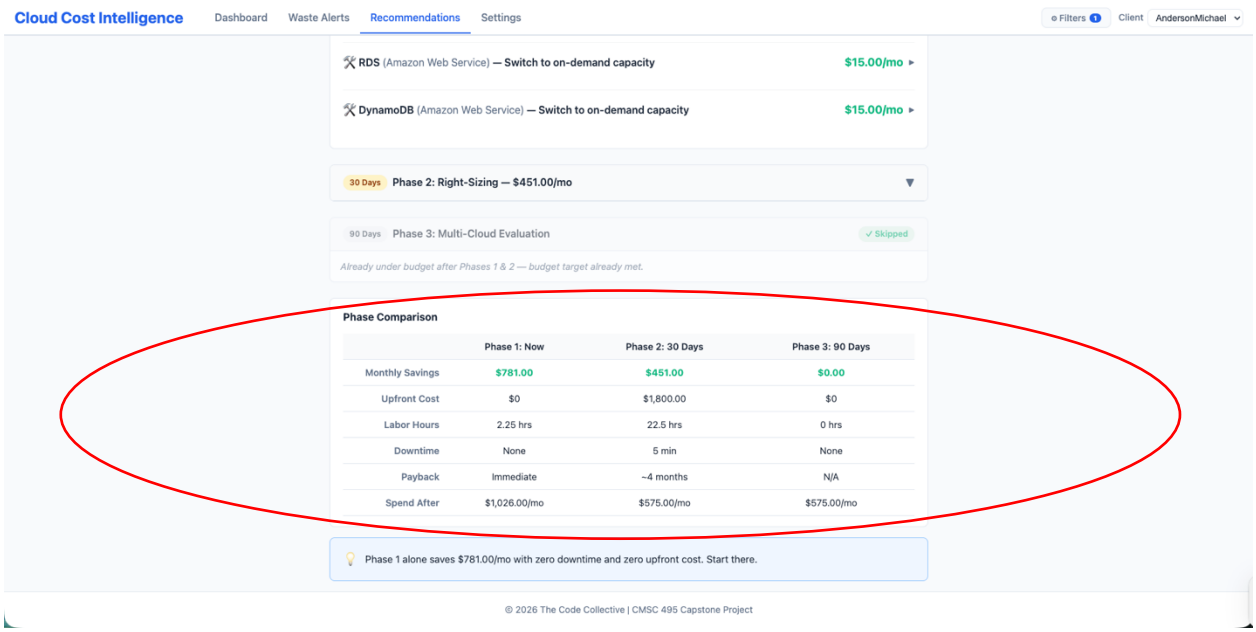

Figure 16: Phase Comparison Table

5.7 Exporting Reports

The export functionality allows users to generate downloadable reports in CSV or PDF formats. Export controls are located on the **Settings** page below the Budget Settings section.

Export Option	Button Label	Description
CSV	Cost Analysis Report (CSV)	Tabular cost data suitable for Excel, Google Sheets, or data analysis tools
PDF	Invoice (PDF)	Professionally formatted invoice-style report with cost summaries and provider breakdowns

To export a report:

- Set the desired date range using the **Date Range** filter in the top navigation bar
- Select a client from the **Client** dropdown if you want client-specific data
- Navigate to the **Settings** page
- Scroll to the **Export Reports** section
- Click **Cost Analysis Report (CSV)** or **Invoice (PDF)**
- The browser automatically downloads the file when complete

CSV export contents:

Column	Description
Date	Usage record date
Provider	Cloud provider (AWS, Azure, GCP)
Service Name	Specific cloud service
Units Used	Resource consumption units
Total Cost	Cost in USD for that record

PDF export contents:

- Title page with date range and generation timestamp
- Cost summary table (Total Spend, AWS Costs, Azure Costs, GCP Costs)
- Detailed usage table (first 50 records with date, provider, service, units, cost)
- Provider breakdown table with percentages
- Page numbers and footer branding

💡 **Tip:** CSV files are smaller and faster to generate. Use CSV for data analysis in Excel or Google Sheets. Use PDF for sharing formatted reports with stakeholders.

⚠️ **Warning:** Larger date ranges (60–90 days) may take a few seconds to process. Do not click the export button multiple times while a report is generating.

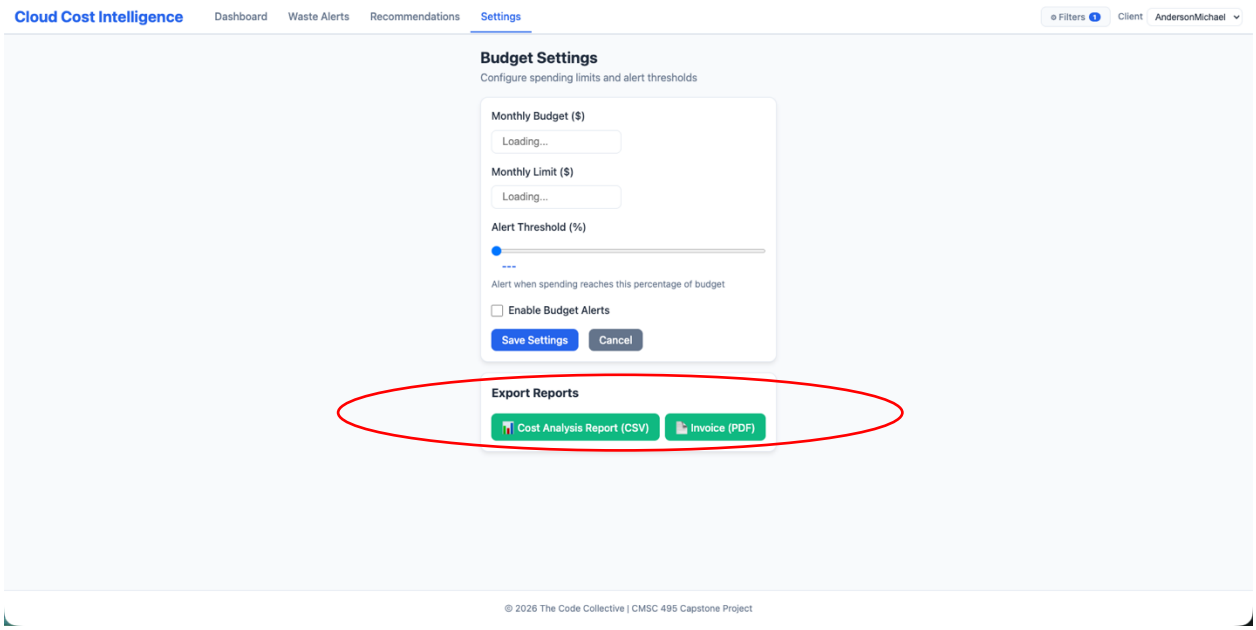

Figure 17: Export Reports section on the Settings page

6. Troubleshooting

6.1 Common Issues

Issue	Possible Cause	Solution
Page shows blank after login	JavaScript disabled or browser cache	Enable JavaScript; clear browser cache and hard refresh (Ctrl+Shift+R)
"Login timeout expired" error	Azure SQL firewall blocking your IP	Find your public IP with `curl ifconfig.me` and add it to the Azure SQL Server firewall rules (see Section 3.2, Step 6)
Docker containers not starting	Docker Desktop not running or previous containers still active	Ensure Docker Desktop is running. Run `docker compose down` to clear old containers, then `docker compose up --build`
No data on dashboard	Azure SQL firewall blocking IP	Add your IP to Azure SQL Server firewall rules in Azure Portal (see Section 3.2, Step 6)
No data on dashboard	FLASK_HOST set to 127.0.0.1	Edit `src/backend/.env` and change `FLASK_HOST=127.0.0.1` to `FLASK_HOST=0.0.0.0`. Restart Docker.
Device code auth fails	Expired code or wrong Microsoft account	Codes expire after 15 minutes. Restart Docker to get a fresh code. Use your @umgc.edu Microsoft account.
Two device code prompts	Flask debug reloader restarts the process	This is expected behavior. Complete both prompts to fully authenticate (see Section 3.2, Step 4).
CORS errors in console	Accessing backend directly instead of through nginx	Access application via http://localhost:8080, not the backend port (5000) directly
Charts not rendering	Chart.js library not loaded	Check browser console for errors; verify internet connectivity for CDN resources
Slow initial data load	Azure SQL Serverless cold start	First request may take 30–60 seconds for serverless tier warmup; subsequent requests are fast
ODBC Driver build fails	ARM Mac (M1/M2/M3/M4) pulling ARM images	Prefix the build command with `DOCKER_DEFAULT_PLATFORM=linux/amd64 docker compose up --build`
.env file not found error	.env not created in src/backend/	Run `cp src/backend/.env.example src/backend/.env` and fill in database credentials (see Section 3.2, Step 2)

6.2 Error Messages

Error Message	Meaning	Resolution
`sqlalchemy.exc.OperationalError: Login timeout expired`	Backend cannot reach Azure SQL database	Add your IP to the Azure SQL firewall (Section 3.2, Step 6). Verify .env has correct server and database names.
`env file .../src/backend/.env not found`	Docker cannot find the environment file	Create the .env file: `cp src/backend/.env.example src/backend/.env` and fill in credentials

Error Message	Meaning	Resolution
`To sign in, use a web browser to open the page...`	Not an error — device code authentication prompt	Follow the URL and enter the code to authenticate (Section 3.2, Step 4)
`WARNING: This is a development server.`	Not an error — standard Flask development mode message	Safe to ignore. Expected in the development environment.

6.3 Getting Help

For issues not covered in this guide:

- GitHub Issues: https://github.com/michaelallenus217-lang/CMSC495-CloudCost/issues
- Team Contact: The Code Collective — CMSC 495 Microsoft Teams channel

7. User Testing Results

User testing was conducted to validate the clarity, usability, and completeness of this User Guide. Test participants were asked to follow the guide step-by-step in a fresh Docker environment and complete key workflows, including launching the application, navigating the dashboard, applying filters, configuring budget thresholds, and exporting reports.
The purpose of testing was to ensure:

- Instructions were accurate and sequential
- Terminology matched the application interface
- Screenshots aligned with actual UI behavior
- Error scenarios were understandable
- No required setup steps were missing

7.1 Test Participants

Tester	Role	Experience Level	Date Tested
Michael Allen	Project Manager / Tech Lead	Intermediate	02/21/2026
Ishan Akhouri	Frontend Lead	Intermediate	02/21/2026
Bryana Henderson	QA Test Lead	Intermediate	02/24/2026

7.2 Test Methodology

Each tester performed a clean install following only the User Guide instructions, starting from cloning the repository through completing all major workflows. The test environment consisted of a fresh clone of the repository with no prior configuration. Testers documented every point where instructions were unclear, incorrect, or missing.
Workflows tested:

#	Workflow	Sections Covered
1	Clone repository and configure environment	3.1, 3.2 Steps 1–2
2	Build and start Docker containers	3.2 Step 3
3	Complete device code authentication	3.2 Step 4
4	Verify application loads with data	3.2 Step 5
5	Configure Azure SQL firewall	3.2 Step 6
6	Navigate all four views (Dashboard, Waste Alerts, Recommendations, Settings)	4.2, 4.3
7	Apply and clear filters	5.2
8	Configure budget settings	5.4
9	Export CSV and PDF reports	5.7
10	Stop and restart the application	3.2 Step 7

7.3 Findings and Resolutions

#	Finding	Severity	Section
1	`.env.example` ships with `FLASK_HOST=127.0.0.1` which does not work in Docker. Backend unreachable from frontend container.	High	3.2
2	No instructions for device code authentication. User sees terminal prompt with no guidance on what to do.	High	3.2
3	Two device code prompts appear due to Flask debug reloader. Confusing without explanation.	Medium	3.2

#	Finding	Severity	Section
4	Guide referenced http://localhost but application runs on http://localhost:8080.	High	
5	Azure SQL firewall step was described as optional. Firewall configuration is mandatory for first-time setup.	High	3.2
6	Login timeout error (HYT00) appeared with no explanation in the guide.	Medium	7.1, 7.2
7	ARM Mac (Apple Silicon) users get ODBC Driver build failure with no guidance.	Medium	3.2
8	Section 4.1 described a browser-based Sign In button and Entra ID login dialog. No such login page exists.	High	4.1
9	Guide said docker-compose (hyphenated). Current Docker uses docker compose (no hyphen).	Low	3.2
10	Directory name after clone listed as cloud-cost-intelligence-platform. Actual name is CMSC495-CloudCost.	High	3.2
11	First data request took approximately 45 seconds with no feedback to user.	Medium	3.2
12	Guide described four summary cards. Dashboard actually shows five (includes GCP Costs).	Low	5.1

7.4 Summary

User testing identified 12 findings: 5 High severity, 5 Medium, and 2 Low. All findings were resolved prior to final submission. The most critical issues involved incorrect URLs, missing authentication instructions, and an environment variable default that prevented the application from functioning in Docker. Testing confirmed that a user following the corrected guide can complete a full installation and use all platform features without external assistance.

8. Glossary

Term	Definition
AWS	Amazon Web Services — cloud computing platform whose simulated cost data is aggregated and displayed in the dashboard
Azure	Microsoft Azure — cloud computing platform whose simulated cost data is aggregated and displayed in the dashboard
Azure SQL	Microsoft's cloud-hosted relational database service used as the project's data store
Chart.js	Open-source JavaScript library used for giving interactive charts and data visualizations in the dashboard
CORS	Cross-Origin Resource Sharing — a browser security mechanism that restricts cross-domain requests. Managed in this project using nginx proxy configuration.
CSV	Comma-Separated Values — tabular data export format compatible with Excel and Google Sheets
Docker	Containerization platform that packages the application and its dependencies into isolated, portable containers
Entra ID	Microsoft Entra ID (formerly Azure AD) — identity and access management service used for device code authentication
FinOps	Financial Operations — practice of managing and refining cloud costs within organizations
Flask	Python web framework used to build the backend REST API
GCP	Google Cloud Platform — cloud computing platform whose simulated cost data is aggregated and displayed in the dashboard
nginx	Web server used in this project to serve frontend static files
ODBC	Open Database Connectivity — standard API for accessing database management systems. ODBC Driver 18 is needed for connecting to Azure SQL.
PDF	Portable Document Format — formatted export format used for invoice-style reports
REST API	Representational State Transfer Application Programming Interface — an architectural style for web services using HTTP methods (GET, POST, PUT, DELETE) to access and manipulate resources
Rightsizing	Adjusting cloud resource allocations to better match actual usage and reduce costs
Serverless	Azure SQL deployment tier that automatically pauses during inactivity and resumes on demand, causing a 30–60 second cold start on first request
Waste Alerts	Feature that flags underutilized cloud resources based on predefined use thresholds.
Utilization Percentage	Metric calculated as usage divided by service capacity, used to decide waste classification.
Severity Level	Classification (Critical, High, Medium, Low) assigned to waste alerts based on cost impact.

Index