NeoSpend: Daily Expense Tracker with Analytics Dashboard

Submitted By:

Mohammed Sameer

Shaikh Nabeel

Nishith

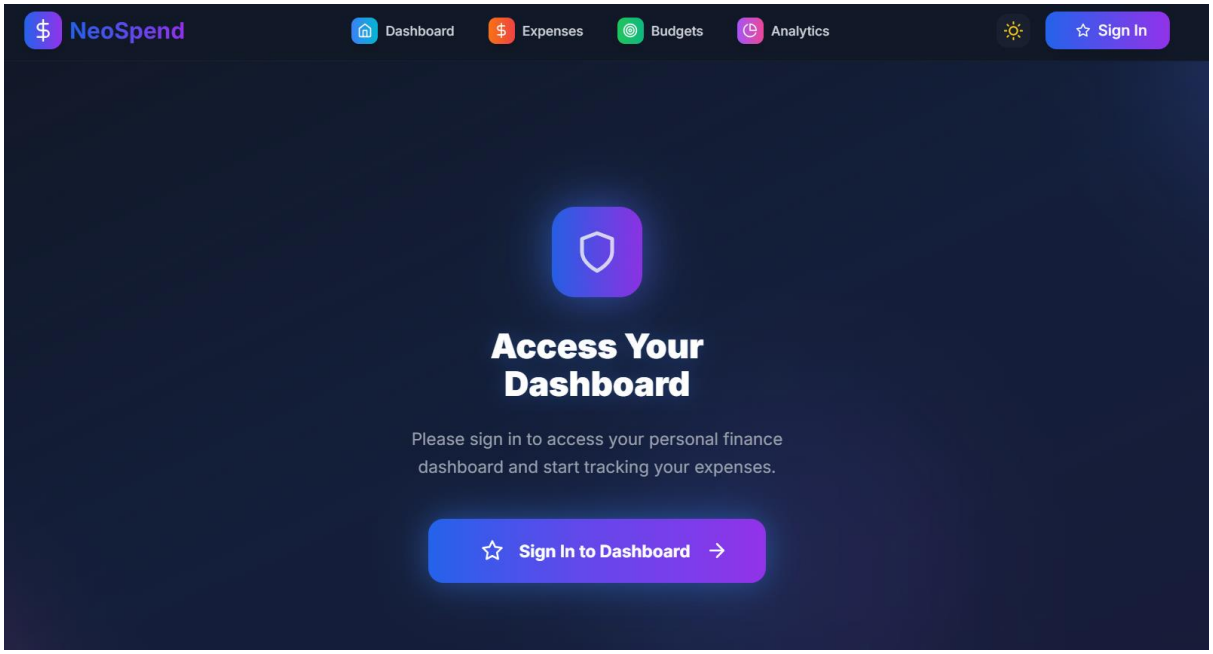
Pranavi Gote

1. Introduction

1.1 Project Overview

NeoSpend – Daily Expense Tracker with Analytics Dashboard – is a modern, serverless web application designed to help individuals manage their personal finances effectively. The system enables users to log daily expenses, categorize them, set budget limits, and visualize spending patterns through interactive charts.

Built on AWS Serverless Architecture with Next.js frontend, the application ensures scalability, cost-effectiveness, and high availability. All user data is securely stored in Amazon DynamoDB with multi-user isolation, while AWS Cognito manages authentication. The frontend is responsive and intuitive, offering a seamless experience across desktop and mobile devices.



1.2 Objectives

The primary objectives of NeoSpend are:

- *To provide a simple and intuitive platform for users to track daily expenses.*
- *To enable categorization of expenses (e.g., Food, Travel, Entertainment, etc.) for better financial insights.*
- *To allow users to set budgets and receive alerts when nearing or exceeding limits.*
- *To offer interactive analytics dashboards for monitoring trends over time (daily, weekly, monthly, annual).*

- *To ensure secure multi-user access with complete isolation of data.*
- *To leverage AWS serverless services for scalability, reliability, and reduced operational overhead.*

1.3 Key Features

NeoSpend offers the following major features:

- *User Authentication & Security*
    - *AWS Cognito for registration and login.*
    - *JWT-based session management.*
    - *Multi-user data isolation.*
- *Expense Management*
    - *Add, edit, and delete expenses with categories, amounts, and notes.*
    - *Filter expenses by date, category, and range.*
    - *Export expenses in JSON format.*
- *Budget Management*
    - *Define budgets per category.*
    - *Real-time progress tracking with visual indicators.*
    - *Alerts when nearing or exceeding spending limits.*
- *Analytics & Insights*
    - *Interactive charts (bar, line, pie) powered by Chart.js.*
    - *Trend analysis: current vs. previous month.*
    - *Category-wise breakdown of expenses.*
    - *Multi-timeframe reports (daily, weekly, monthly, annual).*
- *Responsive Frontend*
    - *Modern UI/UX with Tailwind CSS and animations.*
    - *Dark/Light theme support.*
    - *Mobile-first design for accessibility on all devices.*

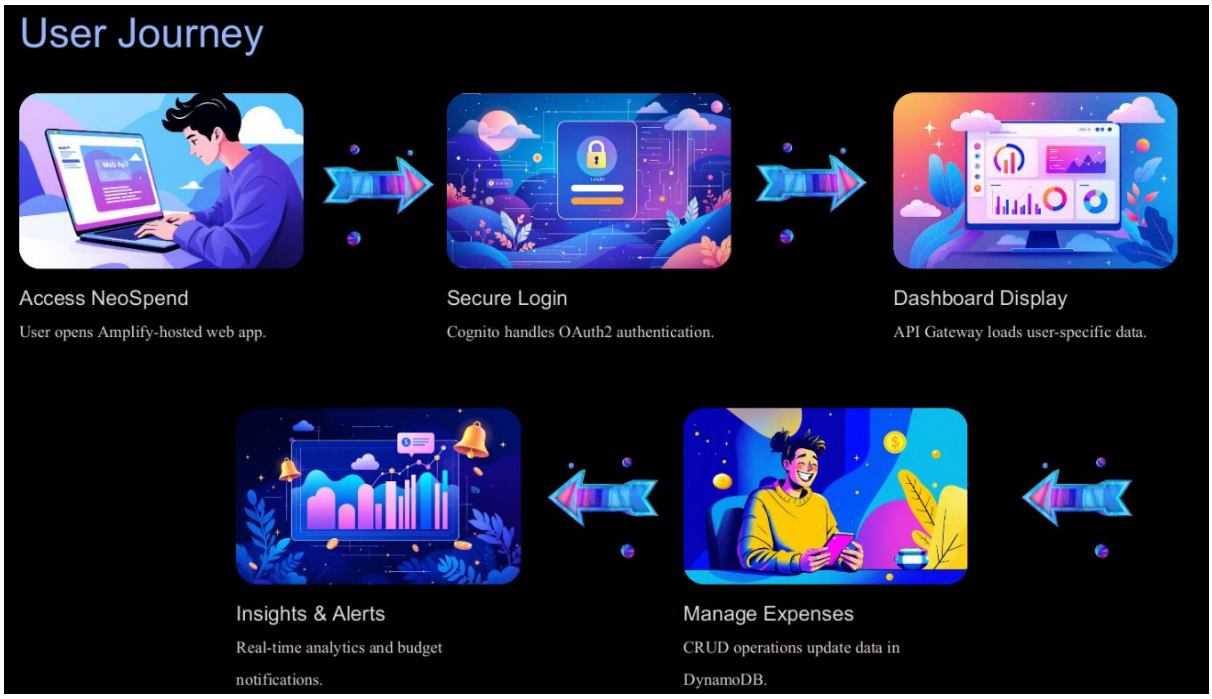
1.4 Scope & Limitations

Scope:

- *NeoSpend is designed as a personal finance tracker with multi-user support.*
- *It can scale seamlessly with AWS Lambda and DynamoDB, making it suitable for thousands of users.*
- *The system focuses on individual users rather than organizational-level finance.*
- *Provides a foundation that can be extended to mobile apps, email/SMS alerts (via AWS SNS), and advanced analytics.*

Limitations:

- *Currently limited to web application only (no mobile app).*
- *Budget alerts and analytics are dependent on real-time data; delays may occur if APIs face downtime.*
- *Export is limited to JSON format (no CSV/PDF integration yet).*
- *Does not include bank account or UPI integration – users must enter expenses manually.*
- *Multi-language and international currency support is not fully implemented (currently optimized for Indian Rupee ₹ and Indian Date/Time).*



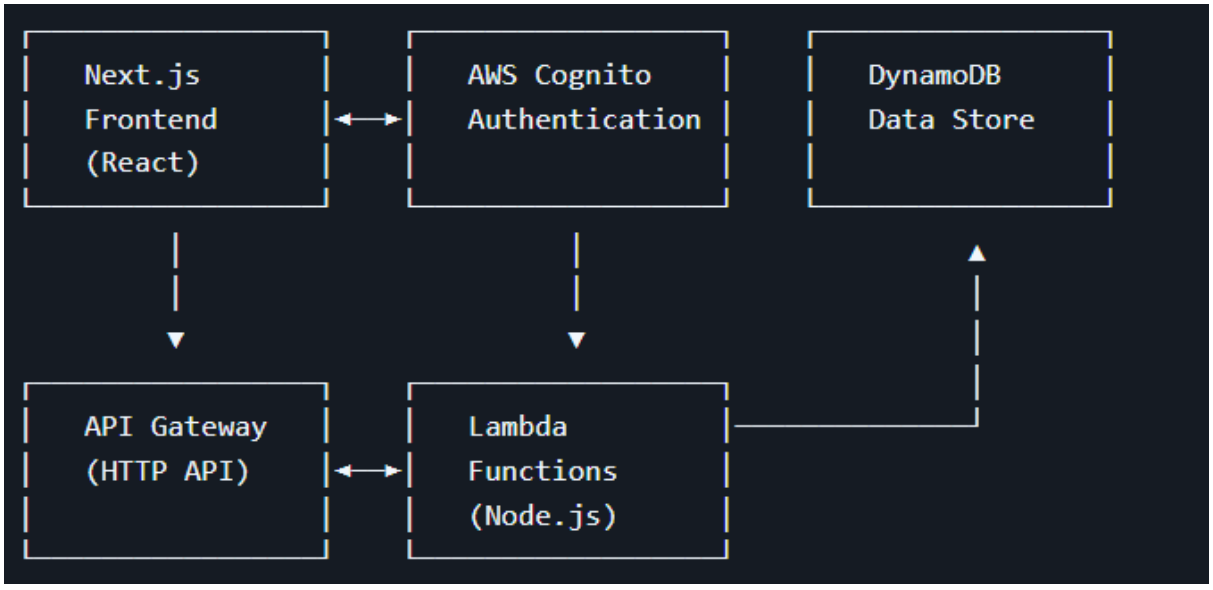
2. System Architecture

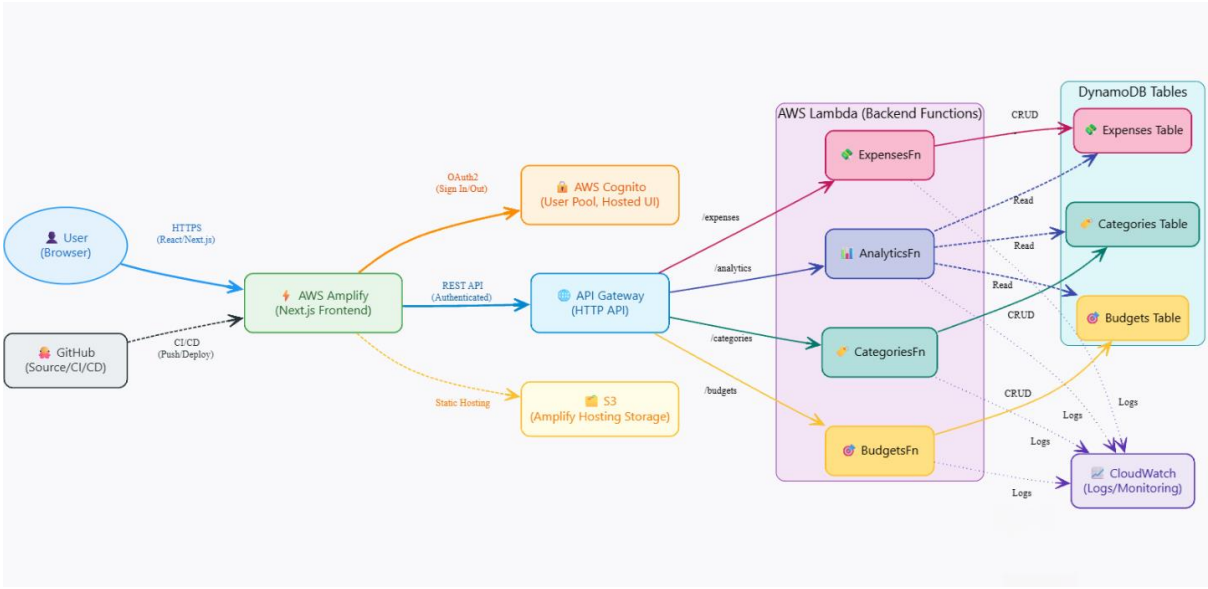
2.1 High-Level Architecture Overview

NeoSpend is designed as a serverless, multi-user personal finance tracker leveraging AWS services for scalability, security, and cost-efficiency. The architecture is divided into four main layers:

1. *Frontend Layer (User Interface) – Built with Next.js (React) and hosted on AWS Amplify for a fast, responsive experience.*
2. *Authentication Layer – Managed by AWS Cognito to ensure secure, multi-user login and session management.*
3. *Backend Layer (Business Logic) – Powered by AWS Lambda functions, exposed through API Gateway for CRUD operations on expenses, budgets, categories, and analytics.*
4. *Data Layer (Storage & Monitoring) – Amazon DynamoDB for persistent, partitioned storage, with AWS CloudWatch for logging and monitoring.*

High Level Architecture





2.2 Component Details

AWS Amplify (Frontend)

- *Hosts and serves the Next.js web application.*
- *Provides CI/CD by integrating directly with GitHub.*
- *Supports SSR (Server-Side Rendering) for dynamic pages.*
- *Automatically rebuilds and redeploys the frontend on every commit.*

AWS Cognito (Authentication)

- *Provides user pool for secure multi-user registration and login.*
- *Offers a Hosted UI for OAuth2-based authentication.*
- *Issues JWT tokens (ID, Access, Refresh) used by the frontend and backend.*
- *Integrates with NextAuth.js for session management in the frontend.*

API Gateway (Routing)

- *Acts as a secure entry point for all API requests.*
- *Validates JWT tokens from Cognito for each incoming request.*
- *Routes requests to appropriate Lambda functions based on path:*
    - */expenses → ExpensesFn*
    - */budgets → BudgetsFn*

    - */categories → CategoriesFn*
    - */analytics → AnalyticsFn*

AWS Lambda Functions (Backend Services)

- *Implements business logic as serverless functions:*
    - *ExpensesFn: CRUD operations for expenses.*
    - *BudgetsFn: Create and track budgets, trigger alerts.*
    - *CategoriesFn: Manage expense categories.*
    - *AnalyticsFn: Aggregate data, generate insights, update charts.*
- *Stateless, lightweight, and scalable on demand.*
- *Integrated with CloudWatch for logging and debugging.*

DynamoDB Tables (Storage)

- *NoSQL database with user-based partitioning.*
- *Tables:*
    - *Expenses Table: Stores expense details (userId, category, date, amount, note).*
    - *Budgets Table: Stores budget records with thresholds.*
    - *Categories Table: Stores user-defined categories.*
- *Supports GSIs (Global Secondary Indexes) for queries by category and date.*

CloudWatch (Logs/Monitoring)

- *Centralized logging system for Lambda executions and DynamoDB events.*
- *Helps monitor system health, identify errors, and optimize performance.*
- *Enables alerting if specific thresholds are crossed (e.g., function failures).*

S3 (Hosting Storage)

- *Stores and delivers static assets of the frontend (HTML, JS, CSS, images).*
- *Integrated with Amplify for global content delivery via CloudFront.*

GitHub (CI/CD)

- *Acts as the version control system for the project.*
- *Every push to the main branch triggers Amplify to build and deploy automatically.*
- *Ensures continuous integration and delivery (CI/CD).*

2.3 Data & Authentication Flow

1. *User Access & Login*
    - *User opens NeoSpend in the browser (Amplify-hosted domain).*
    - *User clicks "Login" → redirected to Cognito Hosted UI.*
    - *Cognito validates credentials and issues JWT tokens.*
    - *Tokens are stored in frontend session via NextAuth.js.*
2. *Expense Logging*
    - *User adds a new expense via the frontend form.*
    - *Frontend sends a POST request with JWT token → API Gateway.*
    - *API Gateway validates the token and routes request to ExpensesFn Lambda.*
    - *Lambda writes data into Expenses Table in DynamoDB.*
    - *DynamoDB confirms write → response sent back → frontend updates dashboard.*

2.4 Security & Scalability Considerations

- *Security:*
    - *Authentication and session management via AWS Cognito.*
    - *All API calls authorized using JWT tokens.*
    - *Data partitioned per user in DynamoDB to ensure privacy.*
    - *Least-privilege IAM roles assigned to each Lambda function.*
    - *HTTPS enforced for all communication.*
- *Scalability:*
    - *Serverless services (Lambda, DynamoDB, Amplify) scale automatically.*
    - *API Gateway supports high throughput with auto-scaling.*
    - *Pay-per-use pricing ensures cost efficiency.*
    - *DynamoDB GSIs optimize queries for large datasets.*

---

3. Database Design

3.1 Design Goals

The database layer of NeoSpend is designed using Amazon DynamoDB, a fully managed NoSQL database service. The primary goals of the design are:

- *Multi-user Isolation*
    - *Each user's data is logically separated using a userId partition key.*
    - *Ensures that no user can access or interfere with another user's financial data.*
- *Scalability*
    - *DynamoDB is horizontally scalable, supporting unlimited records with auto-scaling of throughput.*
    - *Pay-per-request model ensures cost efficiency even as usage grows.*
- *Support for Analytics*
    - *Use of Global Secondary Indexes (GSIs) to optimize queries for analytics (e.g., expenses by date range, category-based breakdowns).*
    - *Efficient read patterns for dashboards and monthly trends.*

3.2 DynamoDB Tables

Expenses Table

- *Purpose: Stores all expense records for each user.*
- *Schema:*

Field	*Type*	*Description*
userId	*String*	*Partition key – unique user identifier (from Cognito).*
expenseId	*String*	*Sort key – unique identifier (UUID) for each expense.*
category	*String*	*Category of expense (e.g., Food, Travel).*
amount	*Number*	*Expense amount in INR (₹).*
date	*String*	*Expense date in ISO format (YYYY-MM-DD).*
note	*String*	*Optional description/note.*

- *Sample Item (JSON):*

```json
{
  "userId": "user123",
  "expenseId": "exp_001",
  "category": "Food",
  "amount": 250,
  "date": "2025-09-10",
  "note": "Lunch at restaurant"
}
```

Budgets Table

- *Purpose: Tracks user-defined budgets for categories and periods.*
- *Schema:*

Field	*Type*	*Description*
userId	*String*	*Partition key – user identifier.*
budgetId	*String*	*Sort key – unique identifier (UUID) for budget.*
category	*String*	*Category assigned to budget.*
limit	*Number*	*Maximum allocated amount.*
period	*String*	*Budget period (daily, weekly, monthly, annual).*
alert	*Number*	*Threshold percentage (e.g., 80%).*

- *Sample Item (JSON):*

```json
{
  "userId": "user123",
  "budgetId": "bud_001",
  "category": "Travel",
  "limit": 5000,
  "period": "monthly",
  "alert": 80
}
```

Categories Table

- *Purpose: Manages user-defined categories for expenses and budgets.*
- *Schema:*

Field	*Type*	*Description*
userId	*String*	*Partition key – user identifier.*
categoryId	*String*	*Sort key – unique category identifier.*
name	*String*	*Category name (e.g., Food, Entertainment).*

- *Sample Item (JSON):*

```json
{
  "userId": "user123",
  "categoryId": "cat_001",
  "name": "Entertainment"
}
```

3.3 GSIs and Access Patterns

To optimize analytics queries and dashboard rendering, Global Secondary Indexes (GSIs) are used:

- *Expenses Table GSIs*
  - *category-index: Query expenses by category for category breakdowns.*
  - *date-index: Query expenses by date for trends and timeframe analysis.*
- *Budgets Table GSI*
  - *category-index: Query budget limits by category.*
- *Access Patterns Supported:*
  - *Get all expenses for a user within a specific date range.*
  - *Get expenses grouped by category for pie charts.*
  - *Get budget utilization per category.*
  - *Aggregate total expenses for a month/year.*

3.4 Sample Data & Queries

Query 1: Fetch all expenses by user and date range

```json
{
  "TableName": "Expenses",
  "KeyConditionExpression": "userId = :uid AND #date BETWEEN :start AND :end",
  "ExpressionAttributeValues": {
    ":uid": "user123",
    ":start": "2025-09-01",
    ":end": "2025-09-30"
  },
  "ExpressionAttributeNames": {
    "#date": "date"
  }
}
```

Query 2: Fetch budget for category “Food”

```json
{
  "TableName": "Budgets",
  "IndexName": "category-index",
  "KeyConditionExpression": "userId = :uid AND category = :cat",
  "ExpressionAttributeValues": {
    ":uid": "user123",
    ":cat": "Food"
  }
}
```

Query 3: Fetch all categories for a user

```json
{
  "TableName": "Categories",
  "KeyConditionExpression": "userId = :uid",
  "ExpressionAttributeValues": {
    ":uid": "user123"
  }
}
```

---

4. API Design & Implementation

4.1 Authentication

All API endpoints in NeoSpend are secured using AWS Cognito + JWT tokens.

- *After login, users receive ID and Access Tokens from Cognito.*
- *Tokens must be included in the Authorization header of every request.*
- *Example:*

Authorization: Bearer <jwt-token>

- *API Gateway verifies the token using the JWT Authorizer before forwarding requests to Lambda functions.*

4.2 REST API Endpoints

NeoSpend provides a RESTful API with four main resources:

/expenses

- *GET /expenses → Fetch all expenses of the logged-in user.*
- *POST /expenses → Create a new expense record.*
- *PUT /expenses/{id} → Update an existing expense.*
- *DELETE /expenses/{id} → Delete an expense.*

/budgets

- *GET /budgets → Fetch all budgets for the user.*
- *POST /budgets → Create a new budget.*
- *PUT /budgets/{id} → Update budget details.*
- *DELETE /budgets/{id} → Delete a budget.*

/categories

- *GET /categories → Fetch all user-defined categories.*
- *POST /categories → Add a new category.*
- *PUT /categories/{id} → Edit category name.*
- *DELETE /categories/{id} → Delete category.*

/analytics

- *GET /analytics → Returns aggregated data for dashboards, including:*
    - *Category breakdown (pie chart).*
    - *Monthly trend analysis.*
    - *Total expenses vs. budgets.*

4.3 Request & Response Formats

Example: Add Expense (POST /expenses)

Request JSON:

```json
{
  "category": "Food",
  "amount": 300,
  "date": "2025-09-10",
  "note": "Dinner with friends"
}
```

Response JSON:

```json
{
  "expenseId": "exp_002",
  "userId": "user123",
  "category": "Food",
  "amount": 300,
  "date": "2025-09-10",
  "note": "Dinner with friends",
  "status": "created"
}
```

Example: Budget Creation (POST /budgets)

Request JSON:

```json
{
  "category": "Travel",
  "limit": 5000,
  "period": "monthly",
  "alert": 80
}
```

Response JSON:

```json
{
  "budgetId": "bud_002",
  "userId": "user123",
  "category": "Travel",
  "limit": 5000,
  "period": "monthly",
  "alert": 80,
  "status": "created"
}
```

Example: Analytics (GET /analytics)

Response JSON:

```json
{
  "totalExpenses": 10500,
  "monthlyTrend": {
    "August": 9000,
    "September": 10500
  },
  "categoryBreakdown": {
    "Food": 4500,
    "Travel": 3000,
    "Entertainment": 2000
  },
  "budgetUtilization": {
    "Food": "90%",
    "Travel": "60%",
    "Entertainment": "80%"
  }
}
```

4.4 Error Handling & Status Codes

NeoSpend follows standard HTTP status codes with descriptive error messages:

Status Code	*Meaning*	*Example Scenario*
200 OK	*Successful request*	*Expense fetched successfully*
201 Created	*Resource created*	*New expense or budget added*
400 Bad Request	*Invalid input*	*Missing required fields*
401 Unauthorized	*Invalid/missing JWT token*	*User not logged in*

Status Code	Meaning	Example Scenario
403 Forbidden	User not authorized	Attempting to access another user's data
404 Not Found	Resource not found	Expense ID not present
500 Server Error	Internal error	Lambda execution error

Error Response Example (401 Unauthorized):

```
{
  "error": "Unauthorized",
  "message": "Invalid or expired token"
}
```

Summary Table: API Endpoints

Resource	Method	Endpoint	Description
Expenses	GET	/expenses	Get all expenses
	POST	/expenses	Add a new expense
	PUT	/expenses/{id}	Update expense
	DELETE	/expenses/{id}	Delete expense
Budgets	GET	/budgets	Get all budgets
	POST	/budgets	Create budget
	PUT	/budgets/{id}	Update budget
	DELETE	/budgets/{id}	Delete budget
Categories	GET	/categories	Get all categories
	POST	/categories	Add category
	PUT	/categories/{id}	Edit category
	DELETE	/categories/{id}	Delete category
Analytics	GET	/analytics	Get analytics summary

---

5. Backend Implementation

5.1 Lambda Functions Overview

The backend of NeoSpend is powered by AWS Lambda, a serverless compute service. Each core module of the application is implemented as an independent Lambda function, triggered via API Gateway. This modular design ensures clear separation of concerns, scalability, and fault isolation.

1. ExpensesFn

- *Purpose: Handles all CRUD operations for expenses.*
- *Operations:*
    - *Create new expense record.*
    - *Fetch expenses (with filters like date and category).*
    - *Update existing expense.*
    - *Delete expense.*
- *DynamoDB Access: Reads/Writes to Expenses Table.*

2. BudgetsFn

- *Purpose: Manages budget definitions and alerts.*
- *Operations:*
    - *Create budgets for categories (daily, weekly, monthly, annual).*
    - *Update and delete existing budgets.*
    - *Track progress against expenses.*
    - *Generate alert flag when limit is exceeded.*
- *DynamoDB Access: Reads/Writes to Budgets Table.*

3. CategoriesFn

- *Purpose: Allows users to manage custom categories.*
- *Operations:*
    - *Create new categories (Food, Travel, etc.).*
    - *Update category names.*
    - *Delete categories.*

- *DynamoDB Access: Reads/Writes to Categories Table.*

4. AnalyticsFn

- *Purpose: Aggregates and processes expense and budget data for insights.*
- *Operations:*
    - *Generate category-wise breakdowns.*
    - *Calculate monthly/annual expense trends.*
    - *Compare current month vs. last month spending.*
    - *Return data for visualizations in Chart.js.*
- *DynamoDB Access: Read-only on all three tables.*

5.2 Business Logic & IAM Permissions

- *Business Logic Flow:*
    - *Each API request is authenticated by Cognito → API Gateway.*
    - *API Gateway routes request to the correct Lambda function.*
    - *Lambda validates input, applies business rules, interacts with DynamoDB, and returns response.*
    - *Example:*
        - *When adding an expense, ExpensesFn checks if the category exists → inserts record into Expenses Table → returns success response.*
- *IAM Permissions:*
    - *Least privilege principle applied.*
    - *Each Lambda has a specific IAM role granting access only to its table:*
        - *ExpensesFn → CRUD on Expenses Table.*
        - *BudgetsFn → CRUD on Budgets Table.*
        - *CategoriesFn → CRUD on Categories Table.*
        - *AnalyticsFn → read-only on all three tables.*
    - *No Lambda has cross-write access to another table, improving security.*

5.3 Logging & Debugging with CloudWatch

- *CloudWatch Logs capture execution details of every Lambda call.*
- *Logs include:*
    - *API Gateway request payloads.*
    - *DynamoDB queries and responses.*
    - *Execution errors and stack traces.*
- *Use Cases:*
    - *Debugging failed requests (e.g., 500 Internal Server Errors).*
    - *Monitoring budget alerts and analytics aggregation.*
    - *Setting alarms for repeated function failures.*
- *Example Log Snippet:*

2025-09-10T10:15:32Z INFO New Expense Added:

{

"userId": "user123",

"expenseId": "exp_002",

"category": "Food",

"amount": 300,

"date": "2025-09-10"

}

5.4 Local Development & Testing (SAM / LocalStack)

NeoSpend backend is developed and tested using AWS SAM (Serverless Application Model) with optional LocalStack for simulating AWS services.

- *Local Development Workflow:*
    1. *Clone repository and navigate to /backend.*
    2. *Build functions:*
    3. *sam build*
    4. *Deploy locally:*
    5. *sam local start-api*
    6. *Test endpoint with cURL/Postman:*

7. *curl http://127.0.0.1:3000/expenses*

- *LocalStack:*
  - *Optional tool for simulating DynamoDB and Lambda.*
  - *Allows testing CRUD operations without AWS costs.*
- *Unit Testing:*
  - *Each Lambda tested with mock events simulating API Gateway requests.*
  - *Example: sam local invoke ExpensesFn --event test-event.json.*

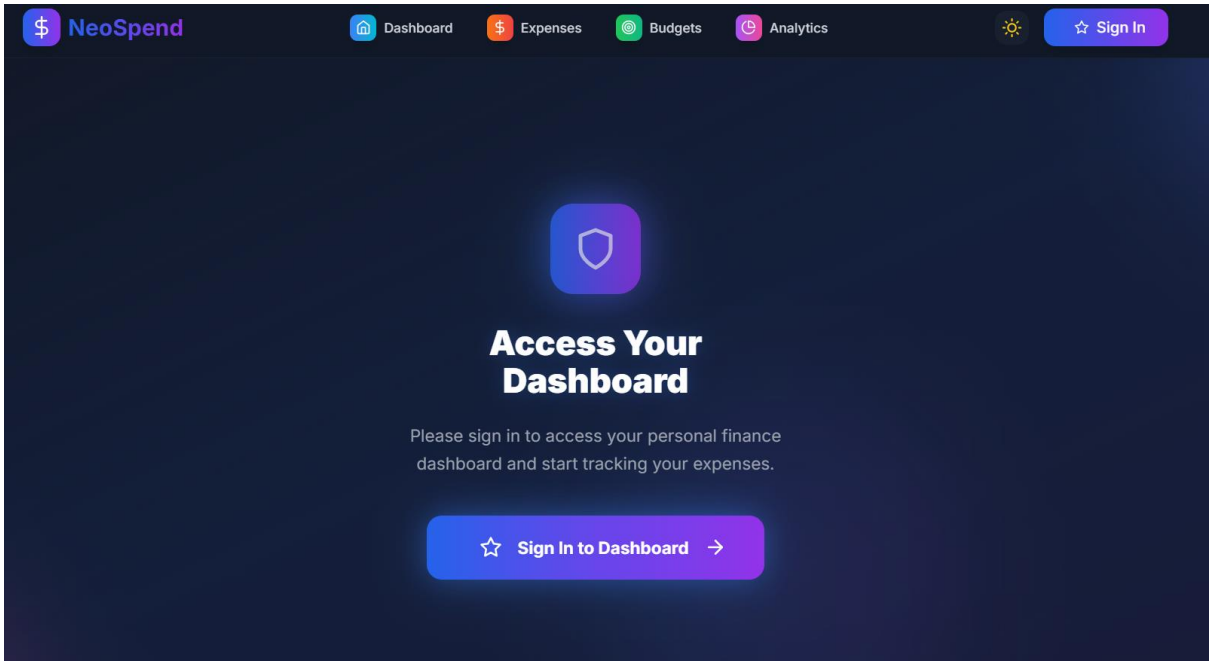
---

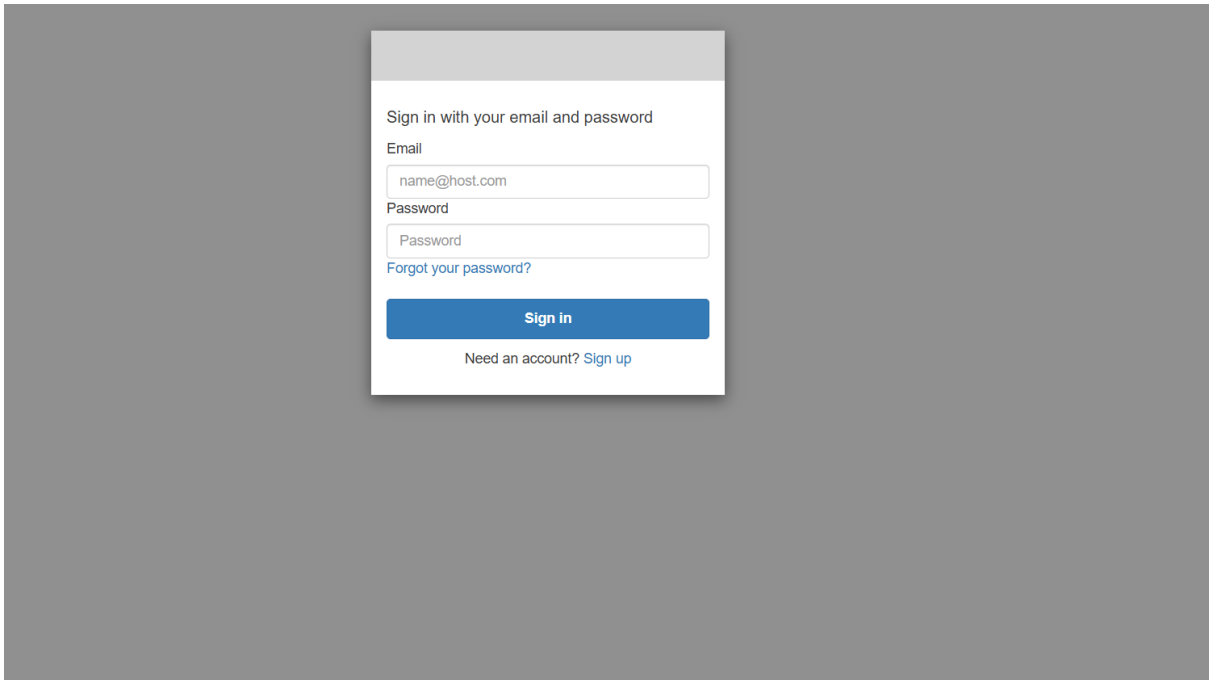
6. Frontend Implementation

6.1 Framework & Libraries

The frontend of NeoSpend is built using Next.js (React framework), ensuring server-side rendering (SSR), static site generation (SSG), and seamless deployment on AWS Amplify. The following key libraries are integrated:

- *Next.js → Provides routing, SSR, and optimized performance.*
- *Tailwind CSS → Utility-first styling for modern, responsive UI with animations.*
- *NextAuth.js → Handles authentication and session management with AWS Cognito.*
- *Chart.js (with react-chartjs-2 wrapper) → Renders interactive visualizations for analytics.*
- *React Hooks → For managing state and API calls.*
- *Axios / Fetch API → For communication with backend APIs.*





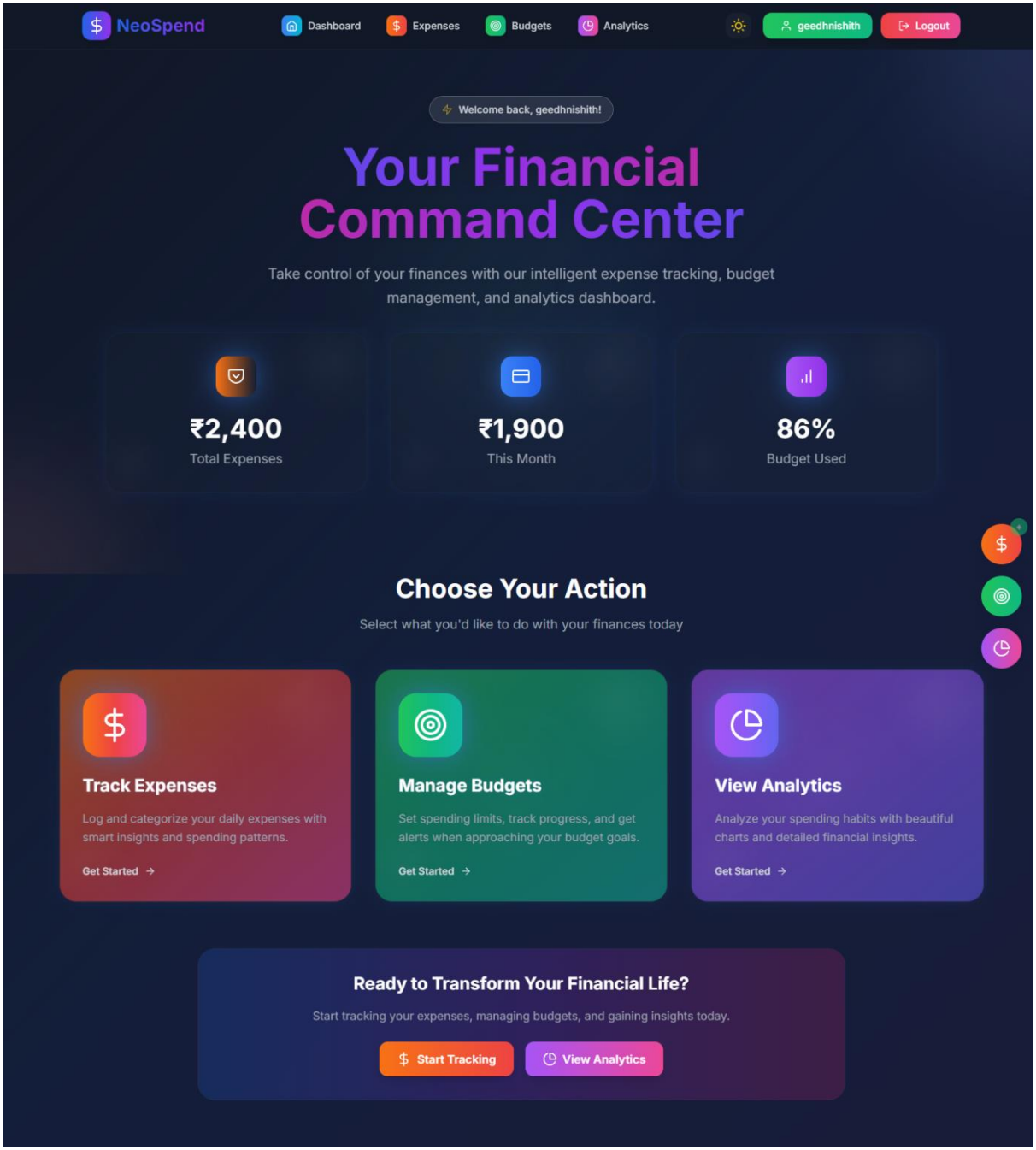
6.2 UI Pages

Login Page

- *Powered by NextAuth.js + Cognito Hosted UI.*
- *Provides secure login and redirects authenticated users to the dashboard.*
- *Supports JWT-based session storage.*

Dashboard

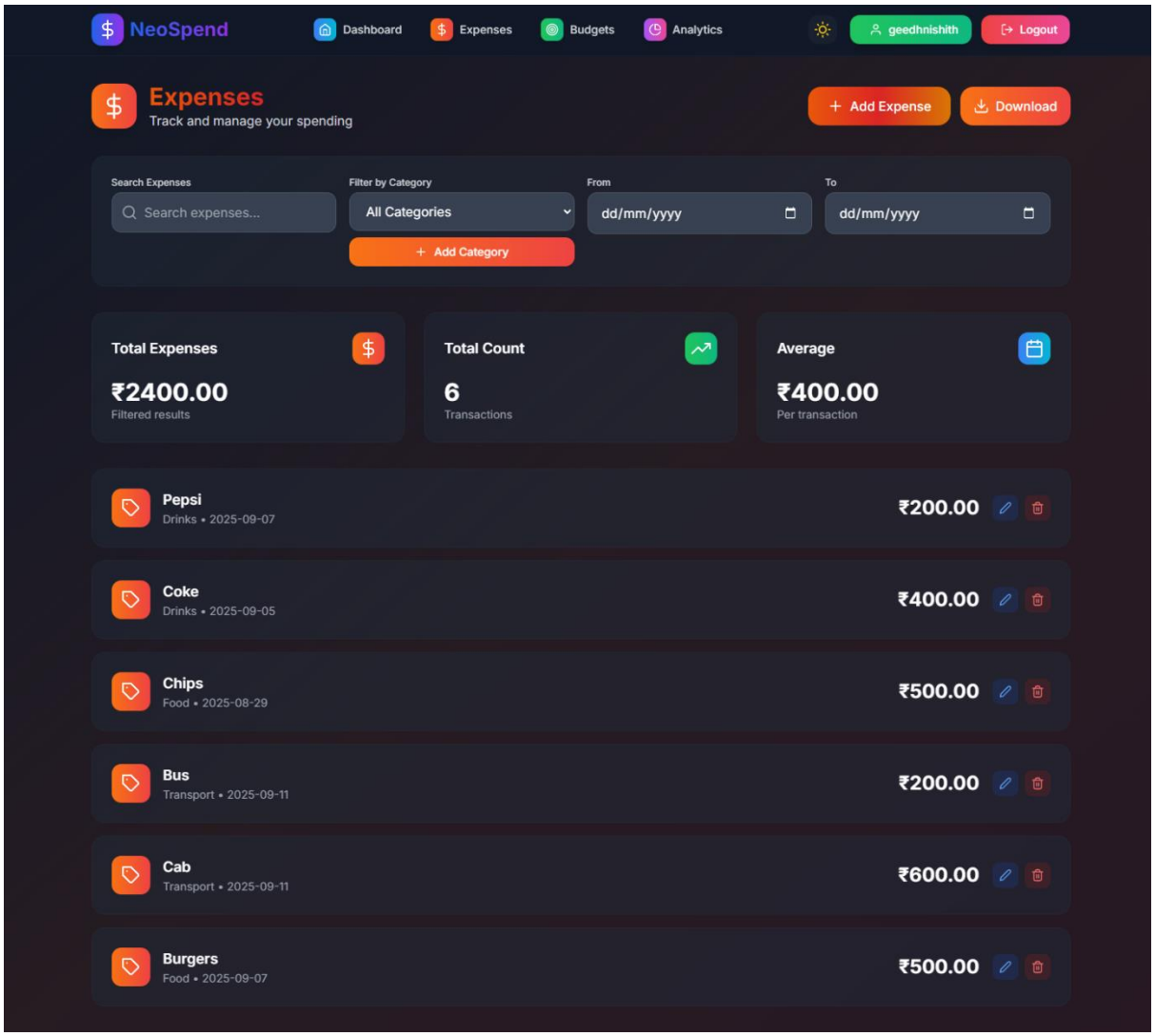
- *Displays summary of expenses and budgets.*
- *Quick links to Expenses, Budgets, and Analytics.*
- *Key metrics like Total Expenses (This Month) and Remaining Budgets.*




NeoSpend
Dashboard
Expenses
Budgets
Analytics
Logout
Take Control of Your Finances
Track expenses, manage budgets, and gain financial insights with our beautiful, intelligent personal finance tracker powered by AWS and modern technology.
10K+
₹2 Crore+
99.9%
Go to Dashboard
Powerful Features
Everything you need to manage your finances effectively, all in one beautiful interface.
Smart Expense Tracking
Budget Management
Analytics Dashboard
Mobile Responsive
Real-time Updates
Cloud Sync
Why Choose NeoSpend?
Experience the difference with our modern, intelligent approach to personal finance management.
Save More Money
Achieve Goals
Better Decisions
What Our Users Say
Real stories from real users who have transformed their financial lives.
Sarah Johnson
Mike Chen
Emily Rodriguez
"NeoSpend helped me reduce my spending habits and save 30% more each month. The analytics are incredible!"
"Finally, a finance app that's both powerful and beautiful. The dark mode is perfect for late-night budgeting sessions."
"The budget alerts have been a game-changer. I never overspend anymore thanks to NeoSpend's smart notifications."
Security & Privacy
Your financial data is protected with enterprise-grade security measures.
End-to-End Encryption
AWS Security
Privacy First
SOC 2 Compliant
Frequently Asked Questions
Everything you need to know about NeoSpend
Is my financial data secure?
Yes! We use bank-level encryption and AWS security services to protect your data. Your information is never shared with third parties.
Can I use NeoSpend on my phone?
How much does NeoSpend cost?
NeoSpend is completely free to use! We believe everyone deserves access to powerful financial tools.
Can I export my data?
Do you offer customer support?
How often is my data backed up?
Your data is automatically backed up in real-time using AWS's secure cloud infrastructure.
NeoSpend
Take control of your finances with our intelligent expense tracker and budgeting platform. Built for modern users who demand both functionality and beauty.
Product
Expenses
Budgets
Analytics
About NeoSpend
Built with Next.js & AWS
Open Source & Free
Privacy First Design
Privacy Policy
Terms of Service
Contact

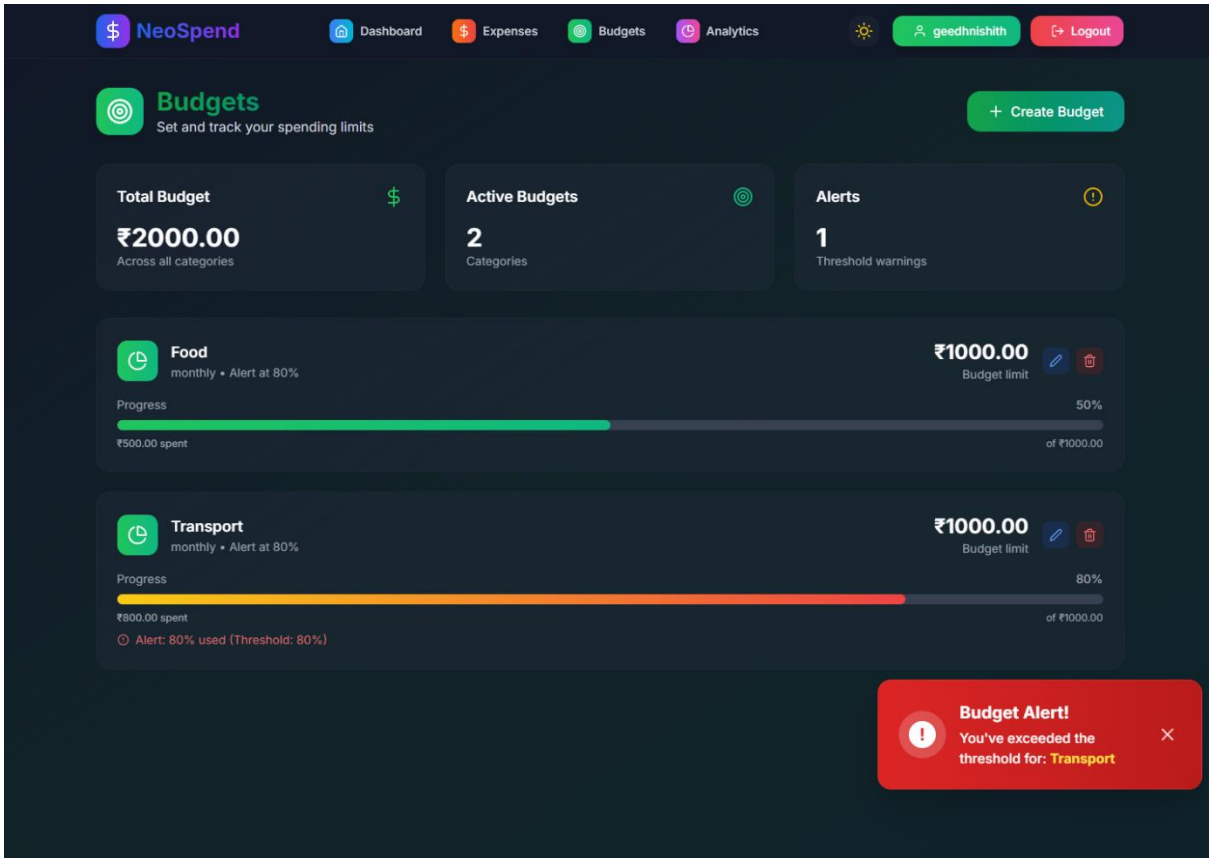
Expenses Page

- *Allows users to add, edit, delete, and filter expenses.*
- *Dropdown for selecting categories.*
- *Calendar picker for selecting date range.*
- *Search and filter options for quick navigation.*



Budgets Page

- *UI for creating and tracking category-based budgets.*
- *Displays visual progress bars for budget utilization.*
- *Alerts when spending exceeds thresholds (red indicators).*



Analytics Page

- *Displays charts and trends for better financial insights.*
- *Views: Weekly, Monthly, and Yearly.*
- *Includes pie chart for category breakdown, line chart for monthly trend, and bar chart for yearly spending.*

NeoSpend Dashboard Expenses Budgets Analytics geedhnishith Logout

Analytics

Insights into your spending patterns

Download

Total Expenses	This Month	Categories	Trend
₹2400.00	₹1900.00	3	+280%
All time	Current month	Active categories	vs last month

Category Breakdown

Daily Analytics

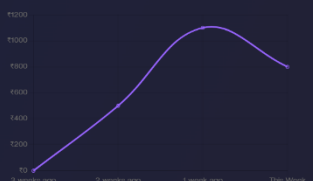
Daily Expenses - Bar Chart

Daily Expenses - Line Chart

Weekly Analytics

Weekly Expenses - Bar Chart

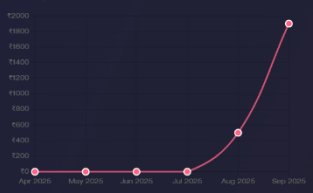
Weekly Expenses - Line Chart



Monthly Analytics

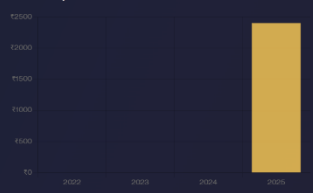
Monthly Expenses - Bar Chart

Monthly Expenses - Line Chart

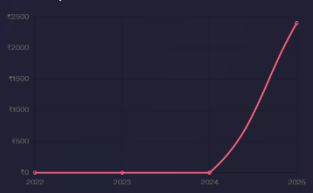


Annual Analytics

Annual Expenses - Bar Chart



Annual Expenses - Line Chart



Top Spending Categories

Food	Transport	Drinks
#1	#2	#3
₹1000.00	₹800.00	₹600.00
2 transactions	2 transactions	2 transactions

Financial Insights

- **Tip:** Start tracking your expenses regularly to get more detailed insights into your spending patterns.
- **Analytics:** The more data you add, the better we can help you understand your financial habits.
- **Goals:** Set up budgets to track your progress and stay on top of your financial goals.

6.3 Chart.js Integration

NeoSpend integrates Chart.js (via react-chartjs-2) for data visualizations.

- *Weekly Trends (Line Chart):*
  - *X-axis: Days of week.*
  - *Y-axis: Daily expenses.*
  - *Shows spending habits over the week.*
- *Monthly Trends (Bar Chart):*
  - *X-axis: Days of the month.*
  - *Y-axis: Expense amounts.*
  - *Highlights overspending patterns.*
- *Yearly Trends (Line/Area Chart):*
  - *X-axis: Months.*
  - *Y-axis: Monthly totals.*
  - *Used for trend analysis and budget forecasting.*
- *Category Breakdown (Pie/Donut Chart):*
  - *Shows percentage spent on each category.*
  - *Helps identify where money is going.*

6.4 Responsive Design & Accessibility

- *Responsive Design:*
  - *Built with Tailwind CSS and mobile-first approach.*
  - *Layout adapts to desktop, tablet, and mobile screens.*
  - *Navigation menu collapses into a hamburger menu on smaller devices.*
- *Accessibility Features:*
  - *ARIA labels for buttons and form inputs.*
  - *High contrast mode for dark/light theme support.*
  - *Touch-friendly interactions for mobile users.*
  - *Keyboard navigation support (tabbing between fields and buttons).*

---

7. Deployment & CI/CD

7.1 Amplify Hosting Setup

The NeoSpend frontend is hosted on AWS Amplify, which provides:

- *Static & SSR Hosting: Builds and serves the Next.js frontend with global distribution via CloudFront.*
- *Build Pipeline: Automates installation, build, and deployment processes.*
- *Environment Variables: Securely stores Cognito client IDs, API Gateway URL, and NextAuth secrets.*
- *Custom Domains (Optional): Supports adding custom domains with SSL certificates.*

Steps followed:

1. *Connected GitHub repository to Amplify Console.*
2. *Configured build settings in amplify.yml (npm install → npm run build).*
3. *Set environment variables (Cognito, API Gateway, NextAuth secret).*
4. *Deployed automatically with every commit.*

7.2 GitHub Integration (Push-to-Deploy)

- *The project is version-controlled on GitHub.*
- *Amplify CI/CD is configured for automatic builds:*

    - *On every push to main branch, Amplify triggers a build → deploy pipeline.*
    - *Developers can test changes in feature branches before merging.*
- *Benefits:*
    - *Eliminates manual deployments.*
    - *Ensures rapid feedback and continuous delivery.*
    - *Maintains deployment history for rollback if needed.*

7.3 Backend Deployment (SAM / Serverless Framework)

The backend (Lambda + API Gateway + DynamoDB + Cognito) was deployed using AWS SAM (Serverless Application Model).

Steps followed:

1. *Navigate to /backend.*
2. *Build and package functions:*
3. *sam build*
4. *sam deploy --guided*
5. *Provided configuration:*
    - *Stack Name: neospend-backend*
    - *Region: ap-south-1*
    - *Tables created: Expenses, Budgets, Categories*
    - *Cognito User Pool and API Gateway provisioned automatically.*

7.4 Rollback & Recovery Strategy

- *Frontend Rollback:*
    - *Amplify maintains previous build history.*
    - *If a new deployment fails, revert to last stable build with one click in Amplify Console.*
- *Backend Rollback:*
    - *SAM deployments are tracked as CloudFormation stacks.*

    - *Previous versions can be restored by rolling back to an earlier stack configuration.*
- *Database Recovery:*
    - *DynamoDB provides Point-in-Time Recovery (PITR).*
    - *This allows restoring tables to any point within the last 35 days.*
- *Disaster Recovery:*
    - *Since the app is fully serverless, scaling and fault tolerance are handled by AWS.*
    - *Logs in CloudWatch ensure traceability in case of failures.*

---

8. Testing & Validation

8.1 Test Strategy

To ensure correctness and reliability, NeoSpend was tested at multiple levels:

- *Unit Testing – Individual functions (Lambda handlers, React components) tested in isolation.*
- *Integration Testing – API Gateway → Lambda → DynamoDB interactions validated using mock/test data.*
- *End-to-End (E2E) Testing – Full workflow tested (login → add expense → track budget → view analytics).*
- *Manual Testing – Conducted using Postman for backend APIs and browser testing for frontend UI.*

8.2 Backend Testing (Lambda + DynamoDB Mock Tests)

- *Tools Used:*
    - *AWS SAM Local – to invoke Lambda functions with mock events.*
    - *Jest + DynamoDB Local/LocalStack – for simulating database operations.*
- *Test Scenarios:*
    - *Add an expense → Validate record stored in DynamoDB.*
    - *Update an expense → Ensure correct expenseId updated.*
    - *Budget creation → Alert flag triggered when threshold exceeded.*

    - *Analytics aggregation → Verify totals, category breakdown, and monthly trends.*

- *Example Unit Test (Jest):*

```
test("Add new expense", async () => {
  const event = {
    body: JSON.stringify({
      category: "Food",
      amount: 200,
      date: "2025-09-10",
      note: "Lunch"
    }),
    requestContext: { authorizer: { claims: { sub: "user123" } } }
  };

  const response = await handler(event);
  expect(response.statusCode).toBe(201);
});
```

8.3 Frontend Testing (Jest, React Testing Library)

- *Tools Used:*
    - *Jest – for component-level unit tests.*
    - *React Testing Library (RTL) – for DOM interaction tests.*
- *Test Scenarios:*
    - *Login page renders correctly.*
    - *Dashboard shows correct totals fetched from APIs.*
    - *Adding a new expense updates the table dynamically.*
    - *Chart.js visualizations render with correct data.*
- *Example RTL Test:*

```
test("renders dashboard heading", () => {
```

```
  render(<Dashboard />);
  const heading = screen.getByText(/Your Monthly Expenses/i);
  expect(heading).toBeInTheDocument();
});
```

8.4 Manual Testing Checklist

Feature	Test Case	Status
Authentication	User can register/login/logout	✅ Pass
Expenses	Add, edit, delete, filter expenses	✅ Pass
Categories	Add/edit/delete categories	✅ Pass
Budgets	Create budget, track progress, alerts fire	✅ Pass
Analytics	Charts display correct monthly trend	✅ Pass
Responsiveness	Mobile/Tablet/Desktop views	✅ Pass
Error Handling	Invalid input returns proper error message	✅ Pass

---

8.5 Sample Test Data

Expenses Table:

```
[
  { "userId": "user123", "expenseId": "exp_001", "category": "Food", "amount": 250, "date": "2025-09-05", "note": "Breakfast" },
  { "userId": "user123", "expenseId": "exp_002", "category": "Travel", "amount": 800, "date": "2025-09-06", "note": "Cab ride" }
]
```

Budgets Table:

```
[
  { "userId": "user123", "budgetId": "bud_001", "category": "Food", "limit": 5000, "period": "monthly", "alert": 80 }
]
```

Categories Table:

[

{ "userId": "user123", "categoryId": "cat_001", "name": "Food" },

{ "userId": "user123", "categoryId": "cat_002", "name": "Travel" }

]

---

9. Monitoring, Security & Cost

9.1 CloudWatch Logs & Metrics

NeoSpend leverages AWS CloudWatch for centralized logging and monitoring of backend services.

- *Logs Captured:*
    - *All Lambda function executions (inputs, errors, and responses).*
    - *API Gateway request/response status codes.*
    - *DynamoDB read/write activity.*
- *Metrics Monitored:*
    - *API Gateway request count & latency.*
    - *Lambda invocation errors and duration.*
    - *DynamoDB read/write capacity usage.*
- *Alarms Configured:*
    - *Triggered when Lambda error rate exceeds 5%.*
    - *Budget alerts when costs approach predefined thresholds.*

9.2 Security Best Practices

NeoSpend follows AWS security best practices to protect user data and ensure compliance.

- *Cognito Authentication:*
    - *Users authenticate via AWS Cognito User Pool.*
    - *JWT tokens used for session validation and API access.*
    - *Multi-user isolation ensures no cross-data access.*
- *IAM Role Restrictions:*

    - *Each Lambda function is assigned a least-privilege IAM role.*
    - *Example:*
        - *ExpensesFn → CRUD only on Expenses Table.*
        - *AnalyticsFn → Read-only on all tables.*
    - *Prevents unauthorized access to unrelated resources.*
- *HTTPS Enforcement:*
    - *All communications occur over HTTPS (TLS encryption).*
    - *API Gateway configured to reject insecure HTTP requests.*

9.3 Cost Estimation

NeoSpend uses a serverless, pay-per-use model, making it cost-efficient for both light and heavy usage.

Scenario	***Notes (requests / data served / storage)***	***Approx. monthly cost (USD)***	***Approx. monthly cost (INR)***
Light	*5,000 req, 10 GB served, 1 GB storage*	***$1.91***	***₹168***
Medium	*50,000 req, 30 GB served, 5 GB storage*	***$7.21***	***₹636***
Heavy	*500,000 req, 200 GB served, 20 GB storage*	***$48.94***	***₹4,313***

9.4 Scaling & Optimization Plan

NeoSpend is designed for automatic scaling through AWS serverless services:

- *Scalability:*
    - *Lambda scales automatically based on request volume.*
    - *DynamoDB auto-scales read/write throughput with GSIs for efficient queries.*
    - *Amplify delivers assets globally via CloudFront CDN.*

- *Optimization Strategies:*
    - *Use on-demand DynamoDB billing for cost savings in low traffic.*
    - *Enable Lambda provisioned concurrency for reducing cold start latency during peak times.*
    - *Implement API Gateway caching for analytics queries to reduce repeated reads.*
    - *Archive old logs from CloudWatch to S3 for cost efficiency.*

---

10. Known Issues & Future Work

10.1 Current Issues Faced

During development and testing, the following technical issues were identified:

- *Categories not updating in DynamoDB*
    - *Newly added categories are not reflected in the Categories Table or are not synced properly across the frontend.*
- *Dropdown issues in Expenses Page*
    - *The category dropdown is not refreshing dynamically when new categories are added, requiring a manual page reload.*
- *Budget alerts not activating*
    - *The progress bars in the Budgets page are not updating when new expenses are added.*
    - *Alerts (threshold warnings) are not triggered as expected.*
- *Analytics trend not refreshing*
    - *Chart.js visualizations in the Analytics page are not updating with the latest data.*
    - *Monthly trend comparisons show stale data.*
- *Multi-user isolation bugs*
    - *Some API calls are mixing user data, indicating an issue with partition key usage in DynamoDB or JWT validation in the Lambda layer.*

10.2 Troubleshooting Steps

Steps taken to address and debug the above issues:

- *Categories Table Issues*
    - *Checked Lambda write permissions for CategoriesFn.*
    - *Verified DynamoDB GSI updates and tested API responses in Postman.*
- *Dropdown Refresh*
    - *Inspected React state management; identified need to refetch categories list after POST.*
    - *Implemented useEffect hooks to re-render UI when categories update.*
- *Budget Alerts*
    - *Logged CloudWatch outputs for BudgetsFn.*
    - *Confirmed that expenses are being recorded, but aggregation logic requires fixes.*
    - *Adding conditional checks to update progress bars dynamically.*
- *Analytics Not Updating*
    - *Verified that AnalyticsFn is reading data but caching issues may delay updates.*
    - *Working on API Gateway caching policies and real-time refresh logic.*
- *Multi-user Bugs*
    - *Re-verified userId partition key in DynamoDB items.*
    - *Ensured JWT claims are being extracted correctly in Lambda.*
    - *Adding integration tests for multi-user scenarios.*

10.3 Future Enhancements

Planned enhancements to improve functionality, performance, and usability:

- *Email/SMS Budget Alerts (SNS Integration)*
    - *Integrate AWS Simple Notification Service (SNS) to send budget alerts via SMS/email.*
    - *Real-time notifications when thresholds are exceeded.*
- *Pre-Aggregated Analytics*
    - *Implement scheduled Lambda functions to pre-compute analytics (daily/monthly aggregates).*

    - *Reduces load on real-time queries and improves chart rendering speed.*
- *Multi-Currency & Localization*
    - *Extend support beyond INR ₹ with auto-detection of currency and locale formatting.*

---

11. Conclusion

11.1 Summary of Achievements

Over the course of this project, our team successfully designed and implemented NeoSpend – a Daily Expense Tracker with Analytics Dashboard using AWS Serverless Architecture. Key accomplishments include:

- *Developed a scalable, serverless backend using AWS Lambda, API Gateway, and DynamoDB.*
- *Implemented secure user authentication with AWS Cognito and NextAuth.js.*
- *Built a responsive Next.js frontend with modern UI/UX using Tailwind CSS.*
- *Enabled expense logging, category management, and budget tracking with real-time updates.*
- *Integrated Chart.js visualizations for analytics, including category breakdowns and monthly/annual trends.*
- *Deployed the system on AWS Amplify with CI/CD from GitHub.*
- *Conducted testing and validation across unit, integration, and manual scenarios.*

11.2 Key Learnings

Technical Learnings:

- *Hands-on experience with AWS serverless ecosystem (Lambda, DynamoDB, Cognito, API Gateway, Amplify).*
- *Implementation of secure multi-user authentication using JWT tokens.*
- *Designing NoSQL schemas with GSIs to support analytics queries.*
- *Integration of Chart.js with React for real-time data visualizations.*
- *Best practices in CI/CD workflows using GitHub + Amplify.*

Teamwork Learnings:

- *Improved collaboration and task delegation across a team of four members.*
- *Importance of regular sync-ups and communication to identify blockers early.*
- *Exposure to Agile-style iterative development within a fixed timeline.*
- *Learned to document issues, request extensions, and communicate progress effectively.*

11.3 Next Steps & Vision

While NeoSpend currently provides a strong foundation for personal finance management, future iterations aim to expand its features and usability:

- *Short-Term Next Steps:*
    - *Resolve known issues (categories sync, budget alerts, analytics refresh).*
    - *Improve multi-user data isolation and caching for analytics.*
    - *Optimize API performance with pre-aggregated queries.*
- *Long-Term Vision:*
    - *Launch a mobile app version (React Native) for Android and iOS.*
    - *Integrate SNS notifications for real-time budget alerts via SMS/email.*
    - *Extend support for multi-currency and localization.*
    - *Add AI-powered financial insights (spending predictions, savings suggestions).*
    - *Evolve into a comprehensive personal finance assistant, helping users make informed money decisions.*
