WHITEPAPER
BudgetWise: Your Financial Advisor
Document prepared by: Group 5E

The following document is a comprehensive whitepaper on the BudgetWise application. This document highlights the following:

In case of any queries not addressed in the documentation, please contact group 5E's team coordinator: Rohini Vellaisamy Senthilkumar .

A. Executive Summary
BudgetWise aims to help with personal finance management by offering a comprehensive cloud-based solution that simplifies budgeting and expense tracking for users. Traditional methods often involve cumbersome manual processes, leading to disorganization and uninformed financial decisions. By leveraging AWS services, BudgetWise provides an intuitive platform equipped with features such as expense management, budget tracking, visual analytics, and timely notifications. This whitepaper outlines the problem statement, our proposed solution, the technologies utilized, associated costs, recommendations, challenges, and the anticipated benefits of BudgetWise.

B. Problem Statement
Traditional methods of budgeting and expense tracking are time-consuming, error-prone, and lack real-time insights. Users face challenges in staying organized, making informed financial decisions, and planning for the future due to the limitations of manual processes and outdated tools. Furthermore, traditional budgeting tools often lack the functionality and flexibility required to meet the diverse needs and preferences of users.

In light of these challenges, there is a pressing need for a modern and comprehensive solution that leverages the power of cloud technology to streamline personal finance

management. Such a solution should offer users a user-friendly platform equipped with advanced features such as automated expense tracking, dynamic budgeting capabilities, visual analytics, and timely notifications.

C. Architected Solution

a. Proposed Solution

Our proposed solution architecture for BudgetWise encompassed AWS services tailored to deliver a seamless and efficient personal finance management platform. Leveraging **AWS Cognito** for secure user authentication, **AWS Amplify** for hosting the frontend (React) of the application, and **AWS DynamoDB** as the database for storing application data, our aim was to provide users with a robust and scalable infrastructure. Additionally, **AWS Lambda** served as the serverless backend of the application, while **AWS StepFunction** facilitated error handling and orchestration of Lambda functions. The integration of **AWS API Gateway** ensured smooth communication between frontend and backend components, while **AWS QuickSight** enabled the creation of visualization analytics embedded within the application. Furthermore, **AWS Forecast** was planned to forecast spending based on past user data, while **AWS SNS** would alert users via SMS on budget exceeds, and **AWS SES** would provide users with monthly spending reports via email.

While our proposed solution architecture remained subject to potential adjustments, the overarching goal of empowering users to manage their finances effectively remained unchanged.
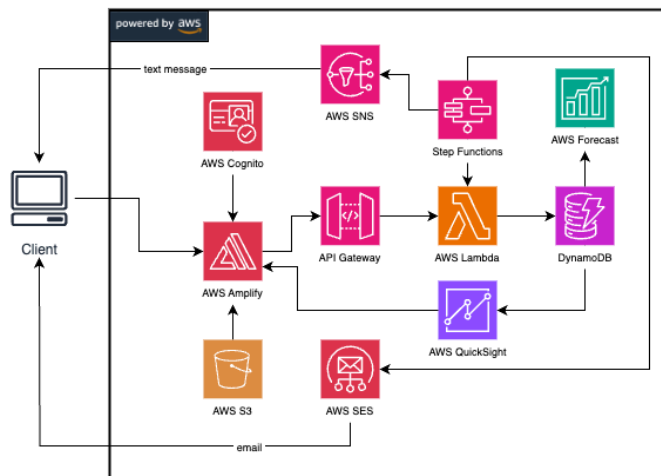
b. **Current Solution**

After conducting spikes and tutorials within the team, we refined our solution architecture, occasionally substituting certain services to better align with the overall architecture. As a result, the current solution architecture includes **AWS EC2** with SG, which hosts the user interface (React) of the application instead of AWS Amplify. Additionally, **AWS Textract** with S3 enables OCR functionality for scanning and storing receipts, enhancing expense tracking capabilities. CloudWatch has been integrated to implement rules for cron-jobs, facilitating daily total calculation for forecasting purposes. During automation, we also had to include and work with **AWS IAM** roles and policies. It's worth noting that most of the other services that were proposed were successfully integrated into the architecture of the BudgetWise solution. However, certain services were dropped (AWS QuickSight and Forecast) due to finding better alternatives, which will be covered in detail in the next section
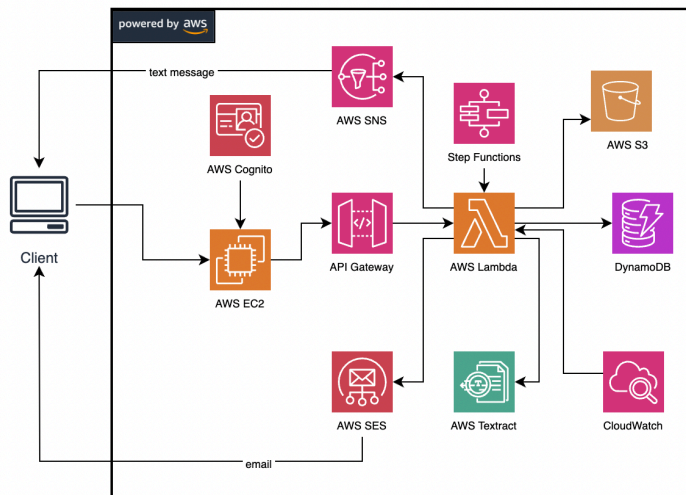
c. **Changes/Pivots – Why?**

The changes made to our solution architecture were primarily driven by several factors, including cost considerations, integration challenges, and comparisons with alternative options. Cost played a significant role in our decision-making process, prompting us to explore more cost-effective alternatives to certain services. By prioritizing these factors, we aimed to ensure the efficiency, effectiveness, and sustainability of the BudgetWise solution architecture. The major changes/pivots in our architecture were:

- **Pivot from AWS Forecast to Python function**: After a spike conducted by the team, the team concluded that implementing a forecasting solution

with Forecast was not feasible due to high cost and time that was required for training the data. Hence, the team decided to pivot to a Lambda hosted Python method that implemented basic forecasting functionality.

- **Pivot from AWS QuickSight to HighCharts**: Due to QuickSight's limited 30-day free trial, it was difficult to maintain continuous development through the course of the project. Noticing that Highcharts provided similar capabilities and analytics (while also being open-source), the team decided to switch to Highcharts.

- **Pivot from AWS Amplify to EC2**: AWS Amplify required an open GitHub repository to provision the frontend. Since by default all the repositories in this course were private, we decided not to circumvent it and pivot to an alternate solution to host our frontend. We opted for using EC2, hosting a docker image of our application,

D. Benefits of cloud technologies

Utilizing cloud infrastructure for the BudgetWise project offers many benefits that significantly enhance its functionality and efficiency. Firstly, scalability is paramount, allowing the application to seamlessly accommodate fluctuating user demands without compromising performance. All of our services are configured to auto-scale based on traffic. Additionally, leveraging tools like Terraform simplified deployment and management processes, ensuring consistency and reproducibility across environments. The robust security features provided by cloud ensured the protection of sensitive financial data. Furthermore, the seamless data synchronization capabilities ensured by cloud platforms facilitated real-time updates and collaboration, enhancing user experience and productivity. The overall automation of processes minimized the need for manual intervention, reducing human error and streamlining operations. Hence, using cloud empowers BudgetWise to deliver a reliable, scalable, secure, and user-friendly solution for effective personal finance management.

E. Estimated Cost:

We approached our solution with a keen awareness of cost considerations that came about from leveraging cloud services. Although most of our services fell under "free tier" during our development phase, the below table shows the estimated cost and resources if one were to run this application at a production level with real-time users:

Service	Assumption	Monthly Cost*
EC2 (t4g.nano)	24/7 usage	$3.024
Lambda	100 requests/day	$0.006
DynamoDB	100GB	$25
Cognito		$0.0055
Textract	5 receipts/day	$0.225
CloudWatch		$0.00001
SNS	10 triggers/month	$0.1 per user

These costs are roughly calculated and will increase if there are more users due to provisioning of bigger compute power.

The overall cost is estimated to be a monthly fixed cost of about $28 (using basic EC2) and a per user monthly cost of about $0.4.

F. Recommendations:

In light of our project experience, we recommend adopting a structured approach to Terraform file organization, emphasizing the separation of files based on specific services. This modularization enhances maintainability and scalability while facilitating easier troubleshooting and updates. Additionally, doubling down on Infrastructure as Code (IaC) automation minimizes manual intervention, ensuring consistency and reliability across deployments. Lastly, consideration of offloading compute power by utilizing Docker images for the frontend, enables efficient resource utilization and enhances performance scalability. These recommendations collectively contribute to a more streamlined, scalable, and cost-effective deployment process for the BudgetWise application.

G. Challenges

During the development phase of our project, we encountered few challenges that required careful consideration and strategic adjustments.. Here are some of the challenges which we encountered during our development:

- **Terraform for automation**: The implementation of Terraform brought about a considerable learning curve, particularly regarding access control and configuration management. Navigating the intricacies of access permissions and configurations within Terraform required careful consideration and

experimentation. Through this process, we gained valuable expertise in managing access policies and fine-tuning configurations.

- **AMI for EC2**: It is only after we finished building and deploying our frontend docker image did we notice that AWS does not provide an AMI for ARM architecture (encountered by default when built with Mac OS) that comes under free-tier. Hence, the team made a collaborative call to go ahead with t4g.nano.
- **AWS SNS for text messages**: This challenge stemmed from a relatively new requirement introduced by AWS, which mandated different provisioning procedures for US-based numbers compared to other regions. Specifically, AWS required the setup of an origination number for long codes, which incurred a cost of $1, while the requirement for a toll-free number demanded a $2 fee. As a result, we conducted our testing with an Indian phone number to mitigate expenses and ensure the smooth functioning of our code within the Sandbox environment. This approach allowed us to validate the functionality of our SNS integration while minimizing costs during the testing phase.

- **AWS SES for emails**: SES requires users to verify their email addresses before they can send emails, a security measure aimed at preventing misuse of the service. While Terraform provides robust infrastructure as code capabilities for automating the deployment of AWS resources, it does not offer a direct means of bypassing the email verification process in the SES Sandbox environment. This presented a challenge for us as we sought to automate the deployment of SES resources without manual intervention.

H. User Guide

To use our application, we created a comprehensive User Guide which can be found in our code repository. This link can be found [here](). This document highlights the following:
- Prerequisites for Demo
- How to run the demo? (Provisioning)
- How to use the application?
- How to clean-up after the demo? (Teardown)

I. Conclusion

In short, BudgetWise transforms personal finance planning with an easy-to-use system, leveraging the latest AWS advancements. This document showcases our framework and how BudgetWise offers forward-looking financial strategies and instant insights, surpassing traditional budgeting methods. We're open to feedback to enhance BudgetWise and better meet our users' needs.