



CSCI 5410

Serverless Data Processing

Sprint 2 Report

Submitted to

Dr. Saurabh Dey

Department of Computer Science

Dalhousie University.

Submitted by Team 4

Alishan Ali

Harsh Vaghasiya

Rutvik Mansukhbhai Vaghani

Sivasubramanian Venkatasubramanian

Ramya Kommalapati

GitLab Repository: https://git.cs.dal.ca/hvaghasiya/csci5410_serverless_f24_group4

Table of Contents

Project Planning Overview	3
Objective	3
Modules Completed.....	3
Testing Performed	4
Module 1: User Management & Authentication Module	5
Module Overview.....	5
Flowcharts Explanation	5
Implementation Details	9
Testing.....	12
Evidence of Testing	12
Module 7: Web Application Building and Deployment	18
Objective:	18
1. Tools and Technologies	18
2. Flow Diagram	18
3. Implementation Details	19
4. Testing and Validation	20
Validation Testing	23
Gantt chart for remaining tasks	24
Meeting Logs.....	25
References:	26

Project Planning Overview

Objective

Sprint 2 focuses on transitioning from project planning to implementation by establishing core components of the system. The emphasis is on completing the foundational modules, validating their functionality, and laying the groundwork for subsequent integrations. The sprint aims to deliver a partially functional system, with critical features like user management and virtual assistance ready for testing and feedback.

Modules Completed

1. User Management & Authentication (Module 1):

- Implemented a robust sign-up and login mechanism with multi-factor authentication (MFA) using AWS Cognito and Lambda [1], [2].
- Stored user data securely in DynamoDB, ensuring compliance with security protocols [4].
- Notifications for successful registration and login were enabled using AWS SNS and SQS [3].
- Validated access control for different user roles (Guest, Registered Customers, and QDP Agents) to restrict and enable functionalities appropriately.

2. Virtual Assistant (Module 2):

- Integrated GCP Dialogflow and Firestore to build a virtual assistant capable of answering basic queries such as registration guidance and site navigation.
- Configured the assistant to fetch results of previously processed files using reference codes.
- Forwarded user concerns to QDP Agents using GCP Pub/Sub, ensuring asynchronous message passing between users and agents.

Testing Performed

- **User Authentication Tests:**
Verified successful registration and login flows for registered users and restricted access for unauthorized users using DynamoDB and AWS Cognito configurations [1], [4].
- **Chatbot Tests:**
Evaluated the assistant's response to valid and invalid queries, ensuring accurate answers and smooth navigation support.

Module 1: User Management & Authentication Module

Module Overview

The **User Management & Authentication Module** ensures secure and seamless access to the QuickDataProcessor system through multi-factor authentication (MFA). This includes three factors:

1. **UserID/Password authentication via AWS Cognito** [1].
2. **Security Question/Answer stored and validated using DynamoDB and AWS Lambda** [2][4].
3. **Mathematical Challenge for additional validation using AWS Lambda** [3].

Two user pools were created:

- **Registered Users Pool** for authenticated customers and agents.
 - **Guest Users Pool** for temporary access with limited features.
-

Flowcharts Explanation

1. User Login Flow

- **Entry Point:** The user opens the app and selects the "Sign In" option.
- **Validation Steps:**
 - Username and password are checked against AWS Cognito [1].
 - Upon successful login, a security question (retrieved from DynamoDB) must be answered correctly [4].
 - Finally, the user solves a randomly generated math challenge (addition/subtraction) [3].
- **Errors:** Errors are displayed for invalid credentials, incorrect answers to security questions, or failed math challenges, allowing retries.
- **Outcome:** Access is granted if all steps are passed; otherwise, errors are shown.

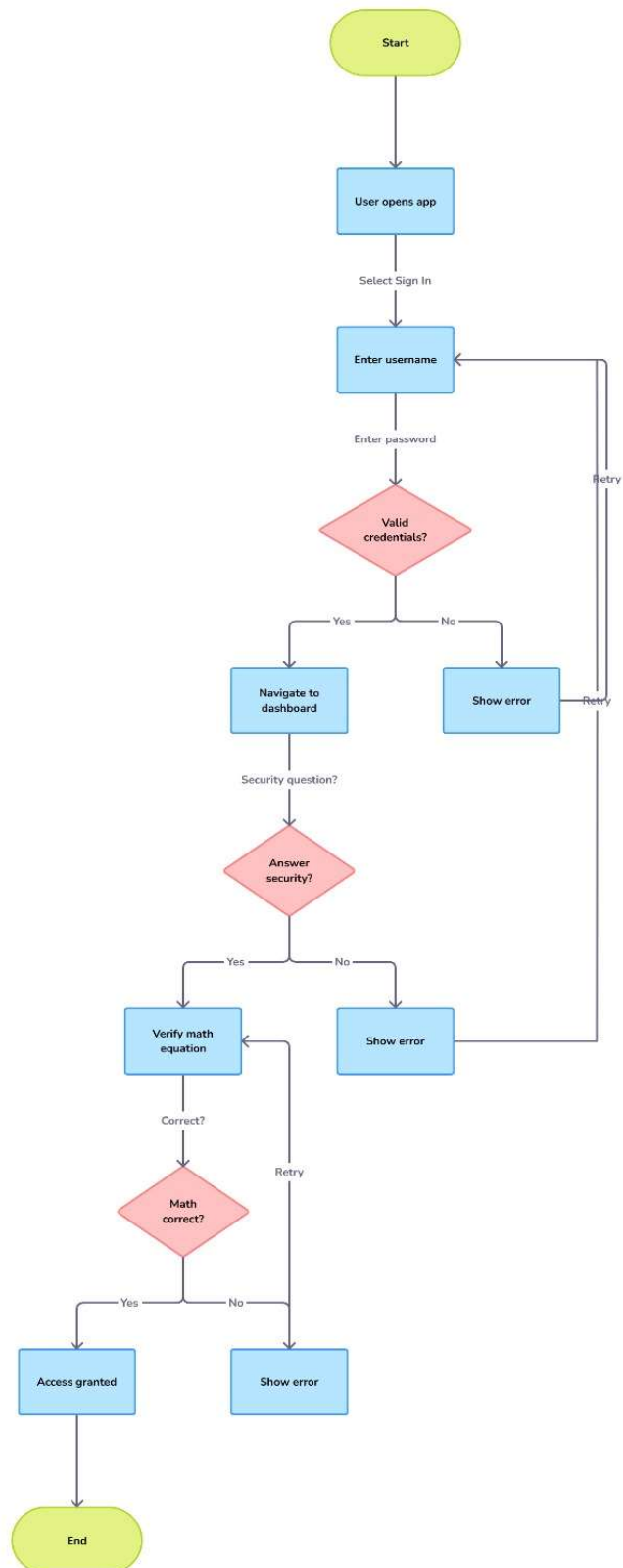


Figure 1 : Login Flow Chart

2. User Signup Flow

- **Entry Point:** The user opens the app and selects "Sign Up."
- **Validation Steps:**
 - The system validates user information submitted via the registration form.
 - If valid, a user is created in AWS Cognito, and a confirmation code is sent [1][2].
 - After entering the confirmation code, the system assigns the user a type and group and stores additional details in DynamoDB [4].
 - Users must answer a security question to complete the signup process [4].
- **Errors:** Errors are shown for invalid data, incorrect confirmation codes, or wrong security question answers, allowing retries.
- **Outcome:** Successful signup registers the user into the appropriate pool.

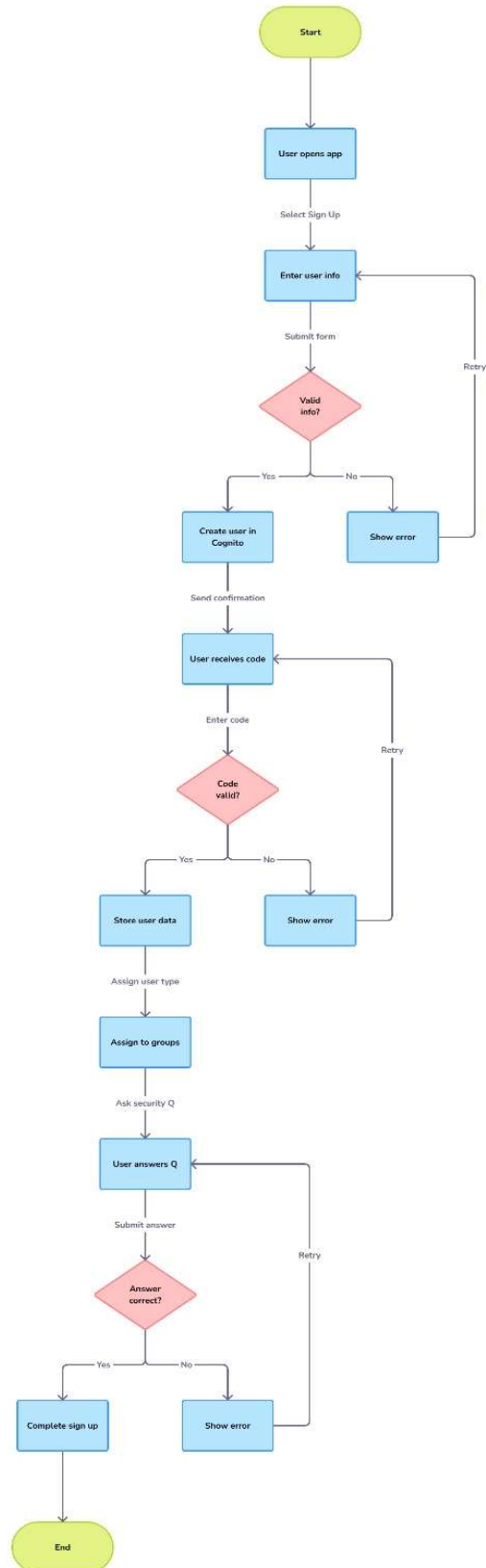


Figure 2: Signup Flow Chart

Implementation Details

1. AWS Cognito Setup

- **User Pools:**
 - **Guest Pool:**
 - Limits users to two file uploads and basic chatbot functionality.
 - Guests are assigned a temporary session with no long-term data storage [1].
 - **Registered Pool:**
 - Provides users with full system access, including unlimited file uploads, virtual assistant usage, and the ability to process and retrieve files [2].
- **Identity Providers:**
 - AWS Cognito supports integration with external identity providers (e.g., Google or Facebook). This can be extended in the future for federated login [1].
- **Password Policies:**
 - Configured with high-security standards: Minimum 8 characters, at least one uppercase letter, one number, and one special character [1].
- **Session Management:**
 - Tokens (ID, Access, and Refresh) are managed via Cognito for secure and seamless session handling [1].

2. DynamoDB Integration

- **Purpose:**
 - Stores user-specific details for multi-factor authentication and role management [4].
- **Indexing:**
 - Global Secondary Indexes (GSI) are configured on the AccountStatus attribute for querying active/suspended accounts [4].
- **Data Encryption:**
 - All sensitive data (e.g., SecurityAnswer) is hashed using a secure algorithm (e.g., SHA-256) before storage [4].

3. AWS Lambda Functions

- **Security Question Validation:**
 - Lambda fetches the user's security question from DynamoDB [3][4].
 - Compares the user's hashed response with the stored hashed answer.
 - Returns success or failure to the authentication workflow [3].
 - Retries are managed up to three failed attempts before locking the account (updates AccountStatus in DynamoDB) [4].
- **Math Challenge Generation and Validation:**
 - A random math problem (e.g., addition or subtraction) is generated by Lambda using the Python random library [3].
 - The problem and its solution are temporarily logged in DynamoDB for session validation [4].
 - On submission, the user's response is compared to the correct solution [3].
- **Logging:**
 - Logs successful and failed attempts in the MathChallengeLogs table.
 - If the user exceeds three failed attempts, the session is terminated.
- **Error Handling:**
 - Implements a try-catch block to gracefully handle DynamoDB unavailability or Lambda timeout [3][4].
 - Sends an error notification to AWS CloudWatch Logs for debugging [3].

4. Notifications

- **AWS SNS:**
 - Sends email notifications for successful registration and login [2][3].
 - Configured to send alerts on failed login attempts after three retries [2].
- **AWS SQS:**
 - Queues notification tasks to decouple the notification service from the core authentication logic, ensuring scalability and fault tolerance [3].

5. Frontend Integration

- **Registration Page:**

- Implements a user-friendly interface using React for collecting username, email, password, and security question/answer.
- Validates user inputs for required fields, password strength, and matching confirmation password [1].
- On submission, the data is sent to an AWS API Gateway endpoint connected to Lambda [3].

- **Login Page:**

- Sequential validation:
 - Collects username and password first [1].
 - If valid, the user is prompted to answer the security question [4].
 - Finally, the math challenge is presented [3].
- Each step submits data to the backend via API Gateway, and the flow proceeds only on success [3].

- **Error Handling:**

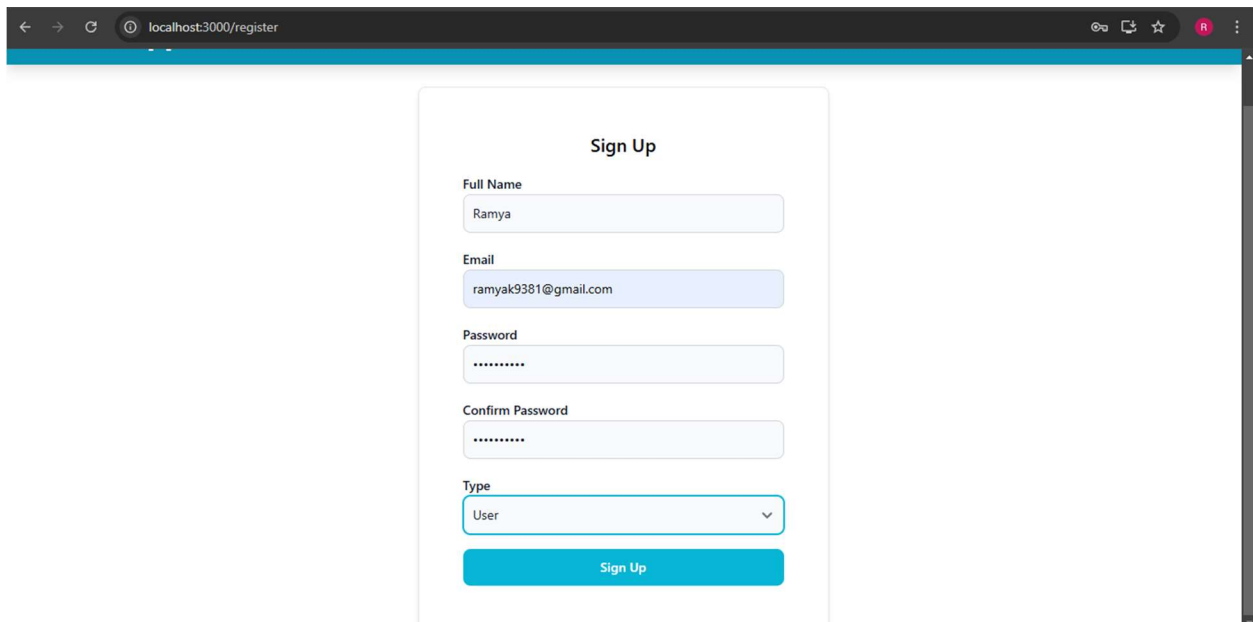
- Error messages are dynamically displayed below form fields for invalid input or failed authentication steps [1].
- Example: "Incorrect security answer. Please try again."

Testing

- **Validation Testing:**
 - **Registration:**
 - Successfully creates users with valid details.
 - Handles invalid inputs gracefully by showing errors [3].
 - **Login:**
 - Passes all three factors of authentication sequentially [4].
 - Prevents access after consecutive failed attempts [3].
- **Performance Testing:**
 - Lambda response time remains under 1 second for generating math challenges and validating responses [3].
- **Error Testing:**
 - Verified the retry mechanism for invalid credentials, incorrect security answers, and failed math challenges [4].

Evidence of Testing

1.Registring into the application with valid details



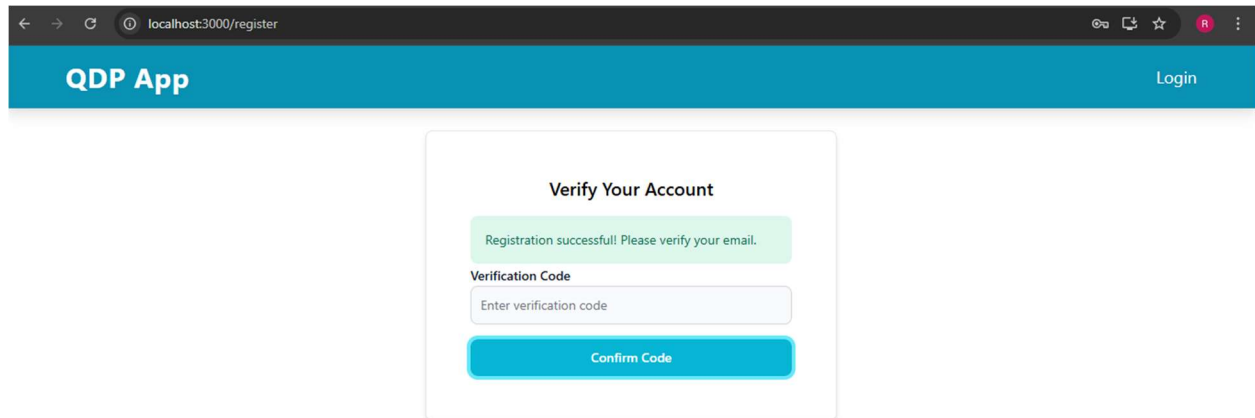
The screenshot shows a web browser window with the address bar displaying 'localhost:3000/register'. The main content area features a 'Sign Up' form. The form has the following fields and values:

- Full Name:** Ramya
- Email:** ramyak9381@gmail.com
- Password:** (masked with dots)
- Confirm Password:** (masked with dots)
- Type:** User (selected from a dropdown menu)

At the bottom of the form is a blue button labeled 'Sign Up'.

Figure 3 : Registering to the application with valid details

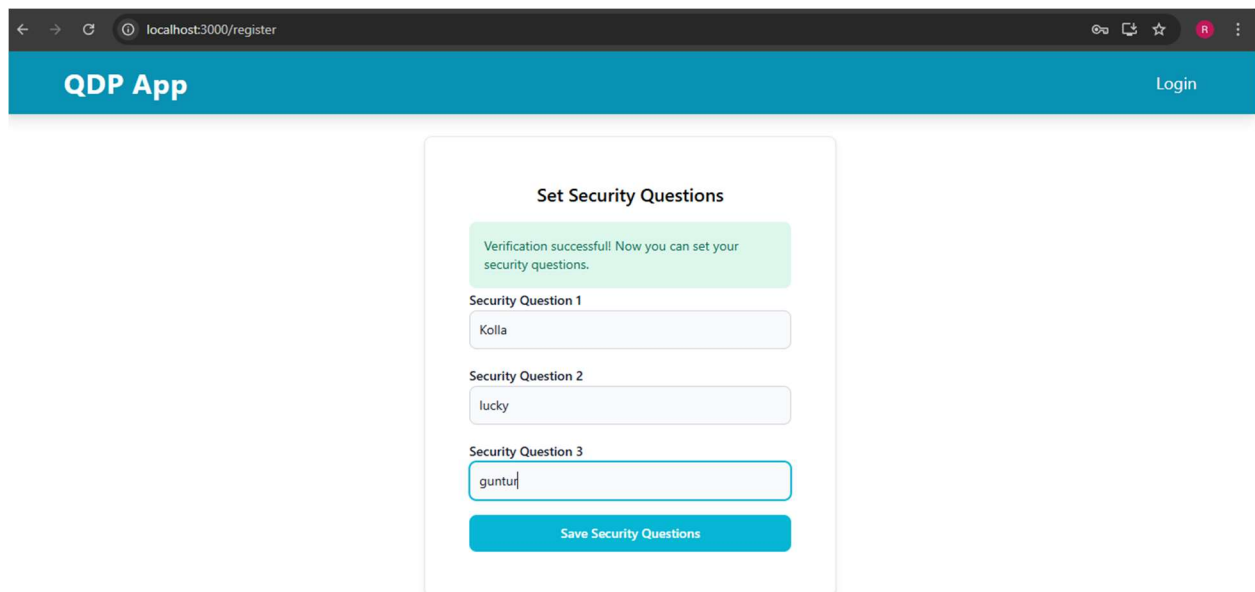
2. Received verification code to mail after signup



The screenshot shows a web browser at localhost:3000/register. The page has a blue header with 'QDP App' on the left and 'Login' on the right. The main content area is titled 'Verify Your Account'. It features a green success message: 'Registration successful! Please verify your email.' Below this is a 'Verification Code' section with a text input field containing 'Enter verification code' and a blue 'Confirm Code' button.

Figure 4 : Verification code sent after successful registration

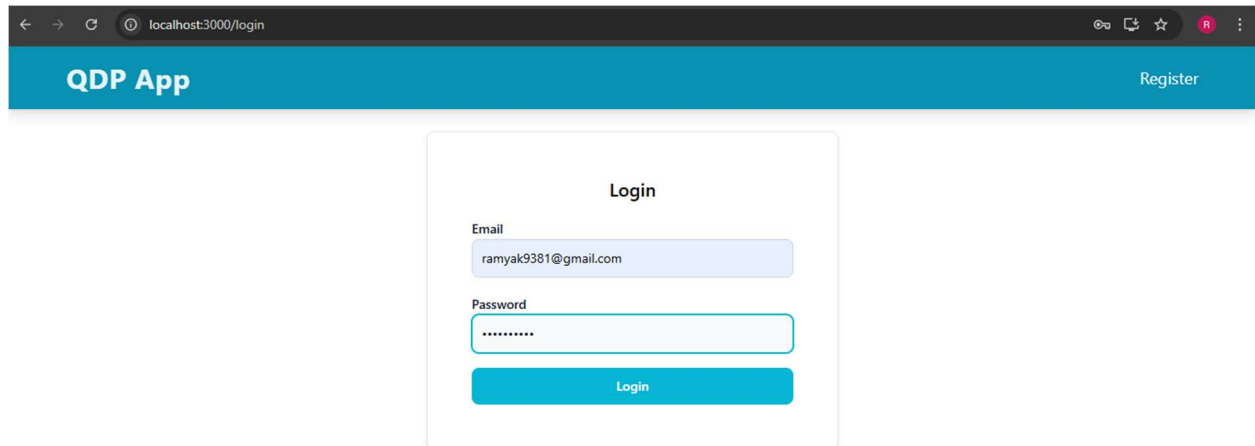
3. Verification code is verified and setting the security questions



The screenshot shows the same web browser at localhost:3000/register. The page has a blue header with 'QDP App' on the left and 'Login' on the right. The main content area is titled 'Set Security Questions'. It features a green success message: 'Verification successful! Now you can set your security questions.' Below this are three 'Security Question' fields. The first field is labeled 'Security Question 1' and contains the text 'Kolla'. The second field is labeled 'Security Question 2' and contains the text 'lucky'. The third field is labeled 'Security Question 3' and contains the text 'guntur'. At the bottom is a blue 'Save Security Questions' button.

Figure 5 : setting the security questions after successful verification of the access code received in email

4.Login with the valid username and password



QDP App Register

Login

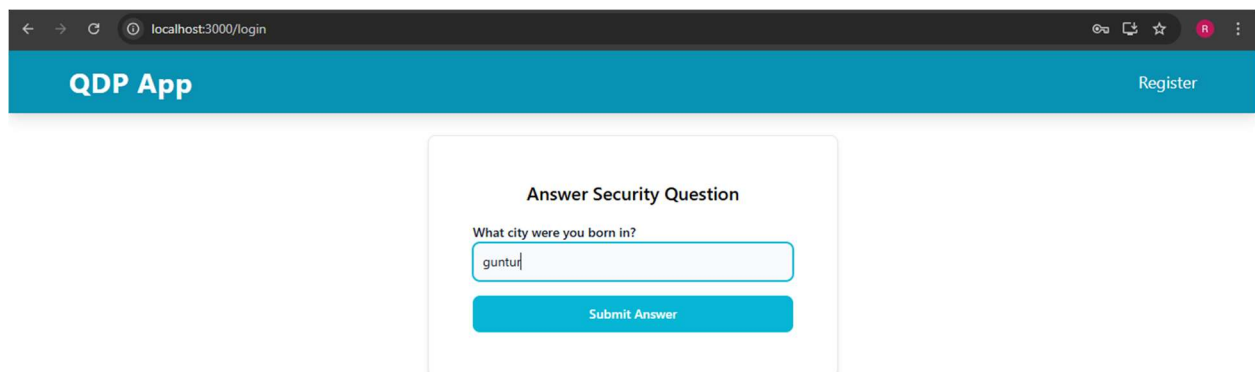
Email
ramyak9381@gmail.com

Password

Login

Figure 6 : Login with valid credentials

5.Ansering the security question after login with valid credentials



QDP App Register

Answer Security Question

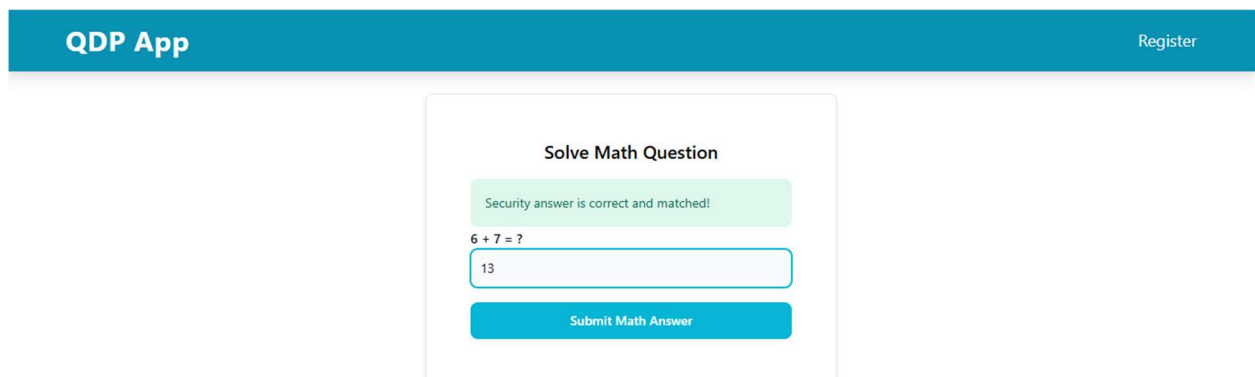
What city were you born in?

guntur

Submit Answer

Figure 2 : Valid security question was answered

6.Solving the math quiz after security question for successful login



QDP App Register

Solve Math Question

Security answer is correct and matched!

6 + 7 = ?

13

Submit Math Answer

Figure 13 : Math Quiz was solved correctly

7. Redirected to dashboard upon answering the security question and solving math quiz correctly

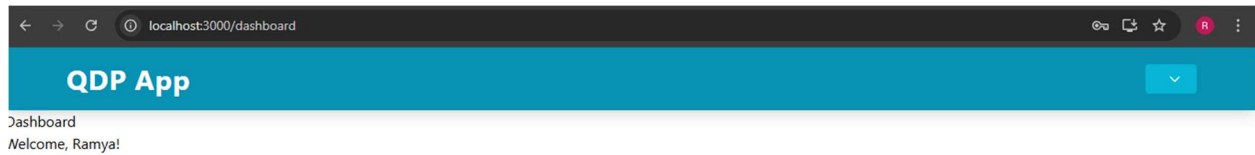


Figure 14 : Dashboard displayed after answering valid security question and solving math quiz correctly

8. Registering to the application with mismatched password i.e invalid details

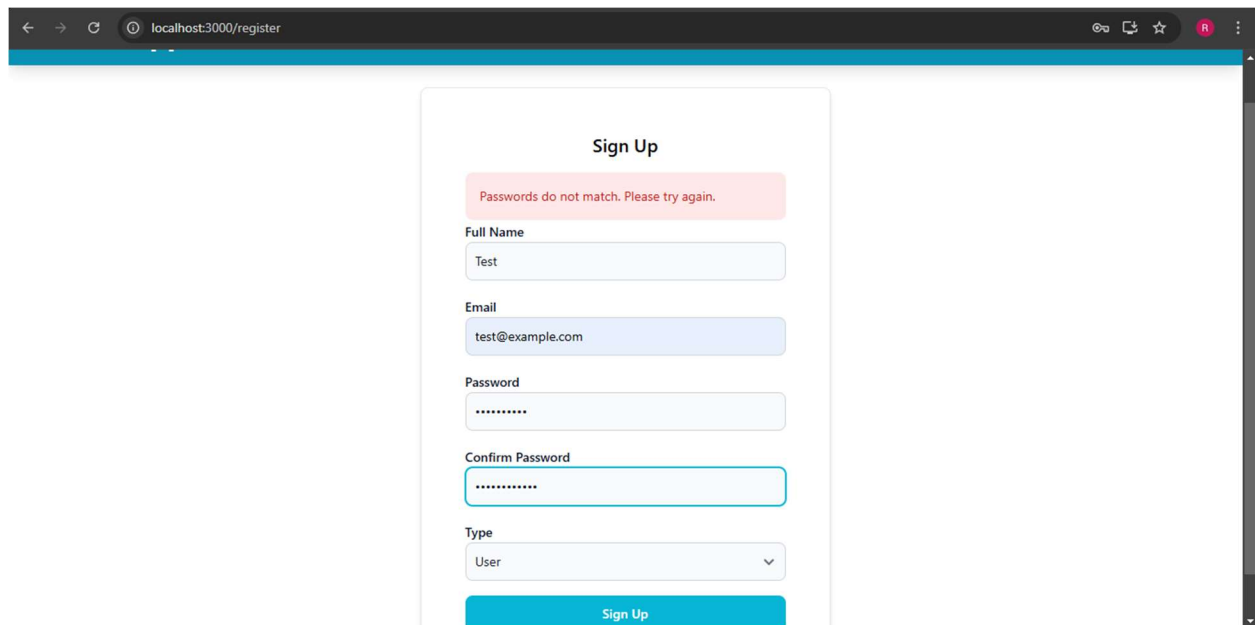
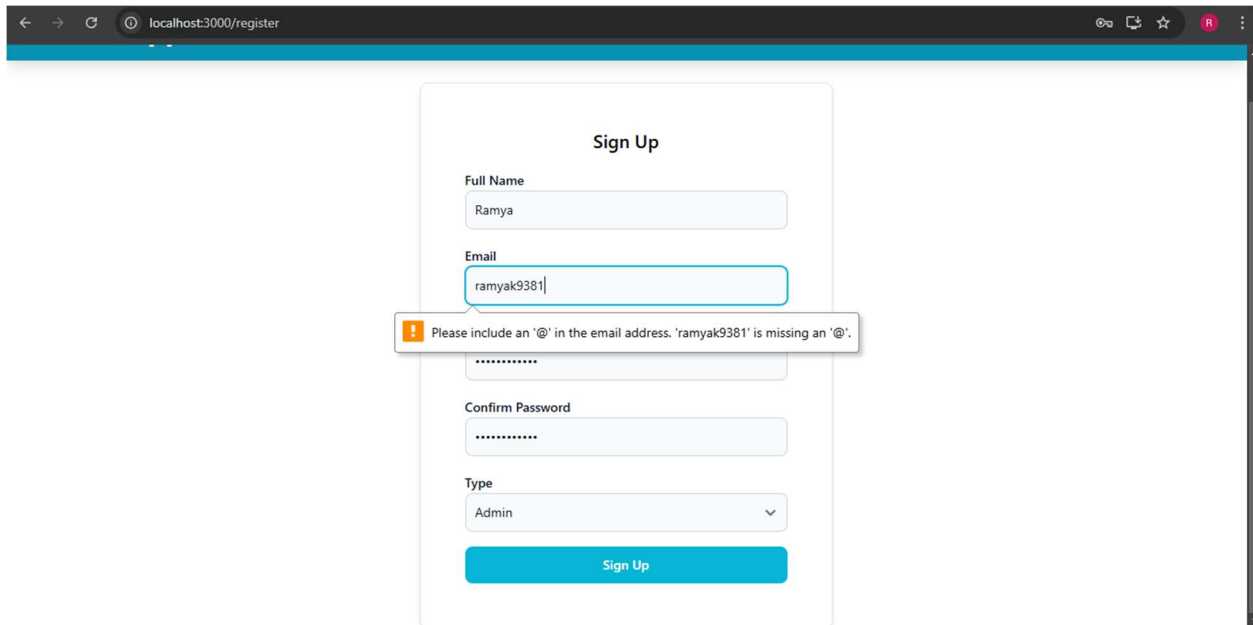


Figure 13 : Registering with passwords not matching to the application

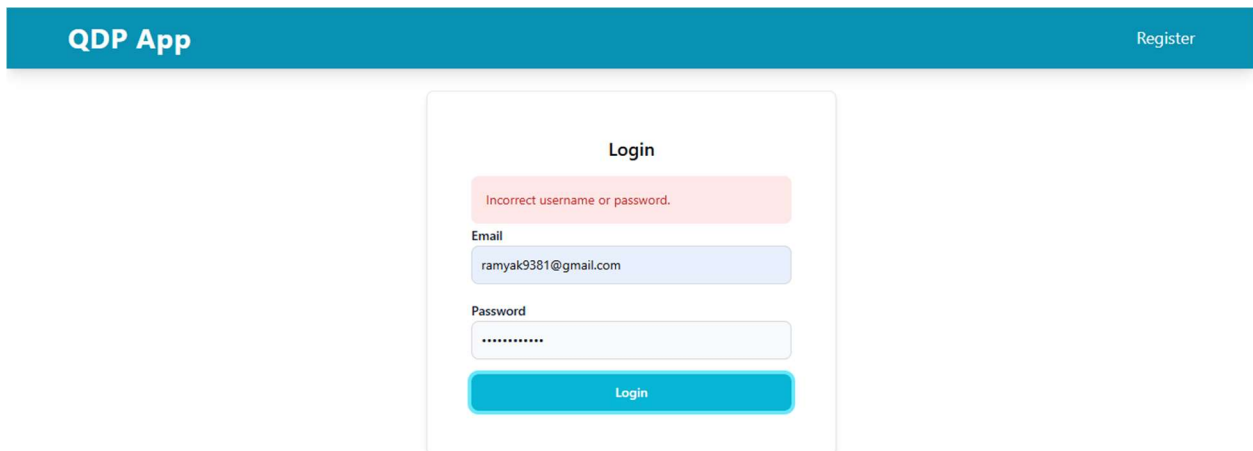
9. Registering into the application using invalid email id



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/register'. The main content area features a 'Sign Up' form. The form includes the following fields: 'Full Name' (containing 'Ramya'), 'Email' (containing 'ramyak9381'), 'Confirm Password' (containing masked characters), and 'Type' (a dropdown menu set to 'Admin'). A blue 'Sign Up' button is at the bottom of the form. An error message is displayed above the 'Confirm Password' field: 'Please include an '@' in the email address. 'ramyak9381' is missing an '@'.' The browser's developer tools are visible on the right side of the window.

Figure 16 : signup with invalid mail id

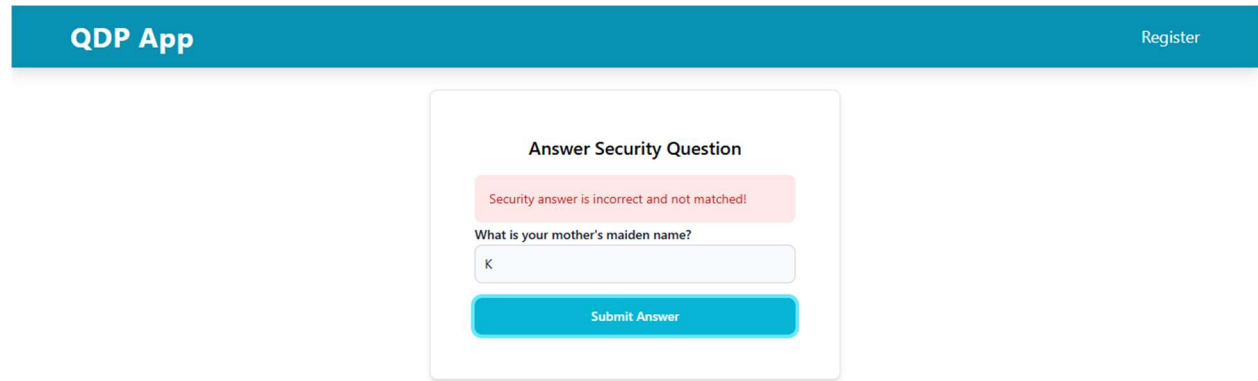
10. Trying to login with incorrect password and username



The screenshot shows a web application interface. At the top, there is a blue header bar with the text 'QDP App' on the left and a 'Register' link on the right. Below the header, there is a 'Login' form. The form has a red error message at the top: 'Incorrect username or password.' Below this, there are two input fields: 'Email' (containing 'ramyak9381@gmail.com') and 'Password' (containing masked characters). A blue 'Login' button is at the bottom of the form.

Figure 17 : Login with invalid credentials

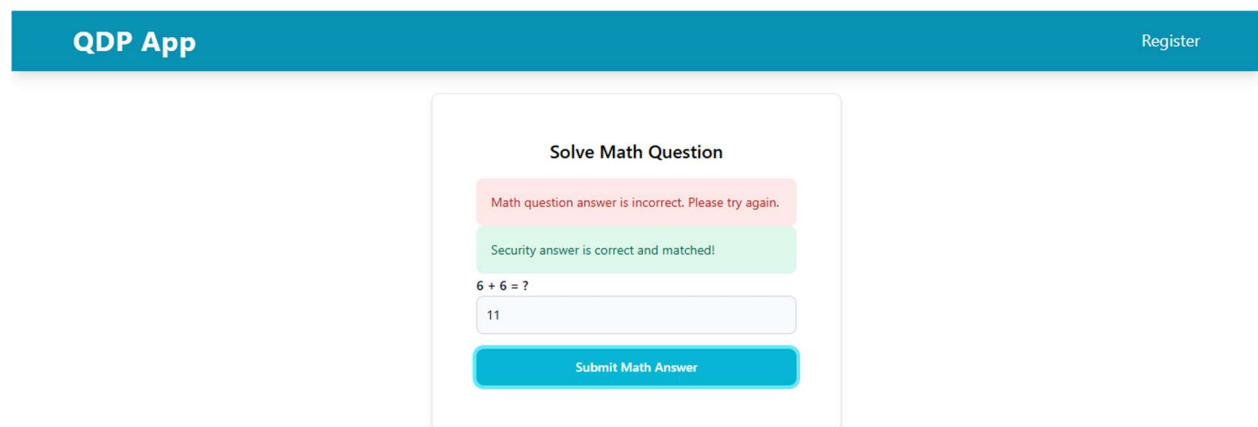
11. Answering the security question wrong after valid credential login



The screenshot shows the 'QDP App' header with a 'Register' link. The main content area is titled 'Answer Security Question'. It displays a red error message: 'Security answer is incorrect and not matched!'. Below this, the question 'What is your mother's maiden name?' is shown. A text input field contains the letter 'K'. At the bottom is a blue 'Submit Answer' button.

Figure 18 : Incorrect or invalid security question after login with valid credentials

12. Answering the math quiz wrong after valid credentials and security question



The screenshot shows the 'QDP App' header with a 'Register' link. The main content area is titled 'Solve Math Question'. It displays two messages: a red one saying 'Math question answer is incorrect. Please try again.' and a green one saying 'Security answer is correct and matched!'. Below these, the math problem '6 + 6 = ?' is shown. A text input field contains the number '11'. At the bottom is a blue 'Submit Math Answer' button.

Figure 14 : Answered incorrect solution to the math quiz after valid credentials, security question

Module 7: Web Application Building and Deployment

Objective:

To build a responsive front-end application using React that interacts with backend services, and to automate the deployment process using CI/CD on Google Cloud Platform (GCP). The final application is hosted on **GCP Cloud Run**, with **Cloud Build** managing the deployment triggered by updates in the GitHub repository.

1. Tools and Technologies

- **Front-End Framework:** React [5].
- **CI/CD Tools:** GitLab, GitHub, GCP Cloud Build [6].
- **Hosting Platform:** GCP Cloud Run [7].
- **Automation Tool:** GCP Cloud Build [8].

2. Flow Diagram

System Architecture Diagram

Below is an overview of the system architecture:

1. **Front-End Application (React):** User interactions occur here, and API calls are made to the backend services [5].
2. **Version Control (GitLab → GitHub):** Code changes are initially pushed to GitLab, then mirrored to GitHub [6].
3. **Cloud Build (GCP):** Automated builds are triggered on GitHub updates, managed through Cloud Build [7].
4. **Deployment and Hosting (Cloud Run):** Cloud Build automates the deployment of the application to Cloud Run, where it is hosted and scaled as needed [8].

Deploy URL: <https://frontend-service-1002147249060.us-central1.run.app/login>

CI/CD Pipeline Flowchart

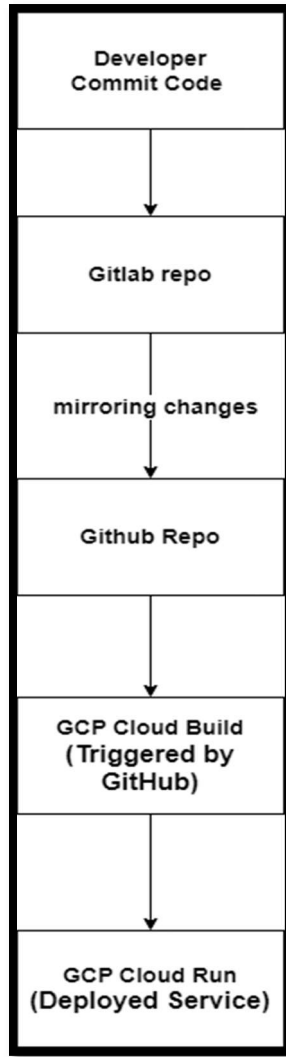


Figure 20: flow Chart of the deployment

3. Implementation Details

Front-End Implementation (React)

The front-end is an application created with React, designed to provide seamless interactions and connect with backend services. Key components include:

- User Registration and Login: Handles user accounts and session management [5].
- Dynamic Data Display: Shows real-time updates based on API responses [5].
- Responsive Design: Ensures compatibility across devices [5].

CI/CD Pipeline Implementation (Cloud Build and Cloud Run)

The CI/CD pipeline is set up to automate deployment through Cloud Build and Cloud Run.

- **GitLab:** Developers commit the code in the GitLab Repository and those changes mirror to the GitHub repository [6].
- **GitHub Repository Trigger:** When a code change is pushed to GitHub, it triggers Cloud Build to start the build process [7].
- **Cloud Build YAML Configuration:** The YAML configuration file (cloudbuild.yaml) includes building steps to install dependencies, run tests, build the application, and deploy to Cloud Run.

4. Testing and Validation

Functional Testing

Update Test: When there is any changes in the frontend and developer commit in the repo it automatically updates the whole deployment.

Testing Proof:

1. GitLab commit done by the developer

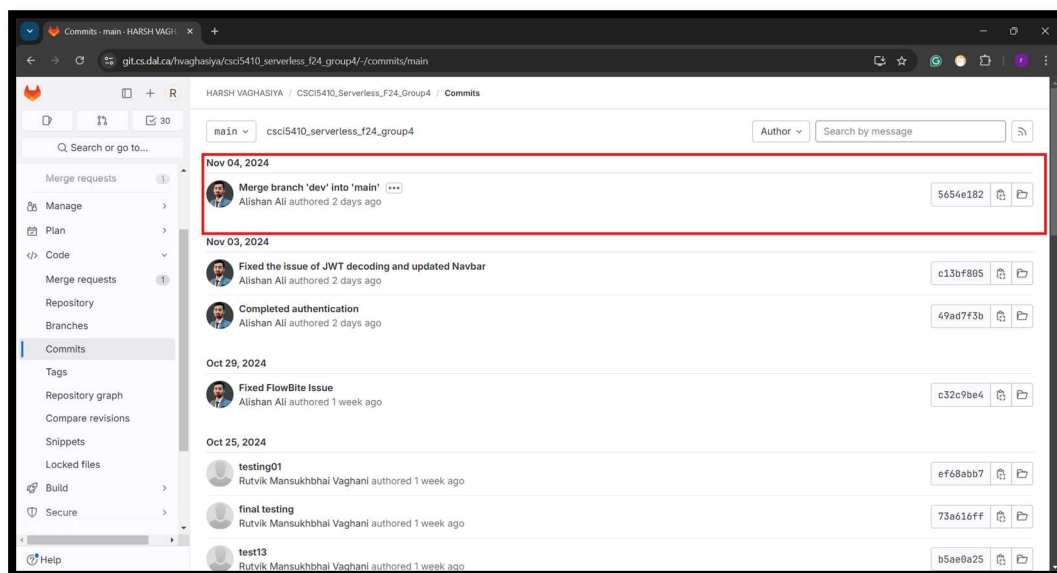


Figure 21: GitLab Commit

2. GitHub Mirror the repository

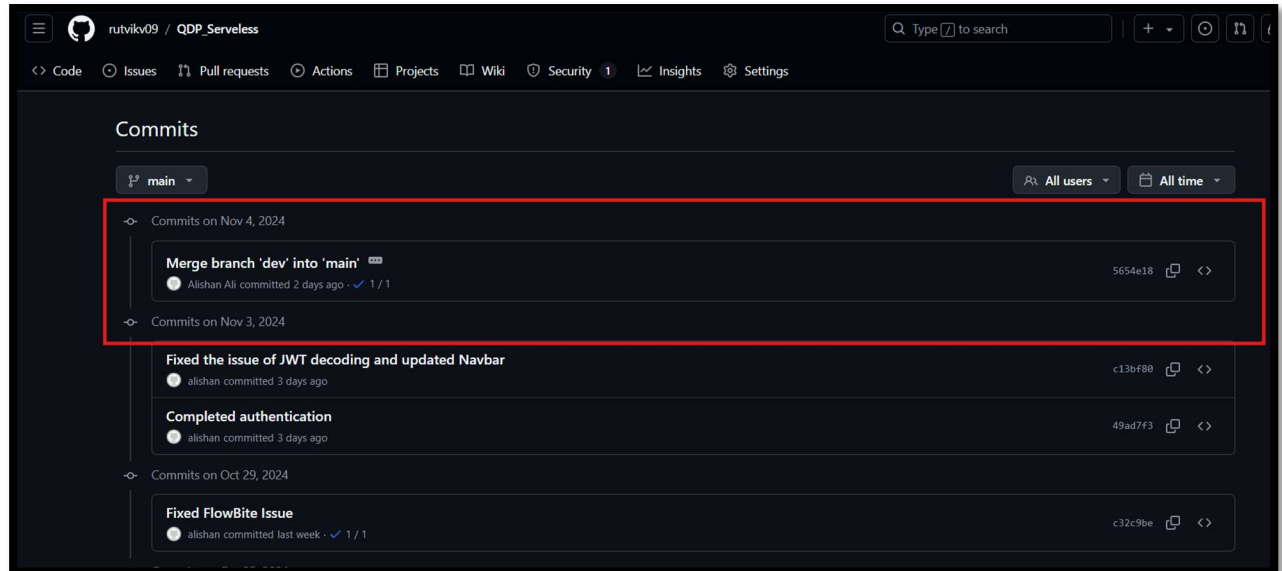


Figure 22 : GitHub Repository Mirror Changes

3. Run the cloud build from the GitHub

Status	Build	Source	Ref	Commit	Trigger name	Created	Duration	Security insights
✓	b7e4feb5	rutvik09/QDP_Serveless	main	5654e18	test1	04/11/2024, 10:34	4 min 31 sec	View
✓	7ae30705	rutvik09/QDP_Serveless	main	c32c0be	test1	29/10/2024, 02:08	5 min 7 sec	View
✓	4185f6ab	rutvik09/QDP_Serveless	main	ef68abb	test1	25/10/2024, 14:58	3 min 30 sec	View
✗	7005a19a	rutvik09/QDP_Serveless	main	ef68abb	test1	25/10/2024, 13:45	3 min 31 sec	-
✓	6ab711d0	rutvik09/QDP_Serveless	main	73a616f	test1	25/10/2024, 13:28	3 min 26 sec	View
✓	2269e9d4	rutvik09/QDP_Serveless	main	b5ae0a2	test1	25/10/2024, 13:10	3 min 28 sec	View

Figure 23: Cloude Build History

4. App Deployed on the CloudeR1un

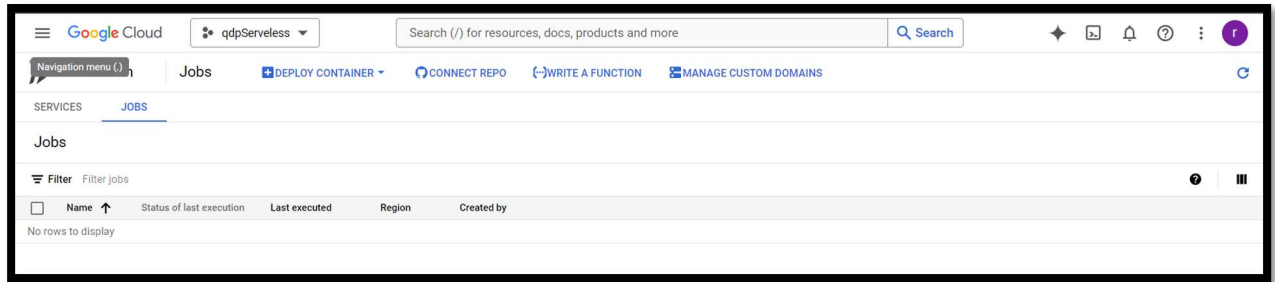


Figure 24 : Cloude Run Deployment

5. Deployed Application with all update changes:

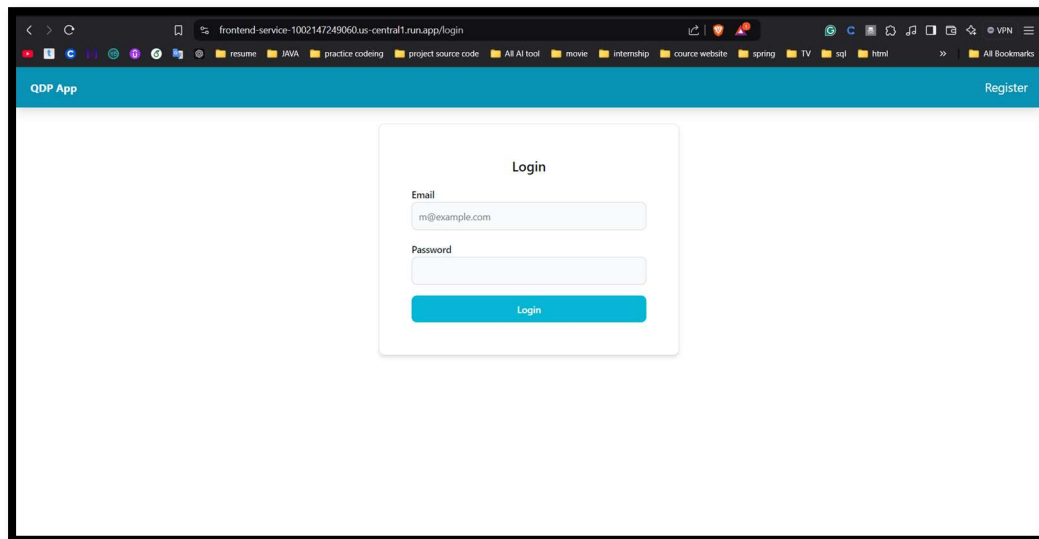


Figure 25: Deployed the Updated APP Frontend

Validation Testing

- Deployment Verification: Each deployment to Cloud Run is validated to ensure it operates as expected without errors or downtimes.
- API Endpoint Tests: Verifies front-end and back-end integration.

Sample Test Cases:

Module	Test Case Description	Expected Outcome	Result
CI/CD Pipeline	GitHub push triggers Cloud Build and deployment	Automatic deployment	Pass

Gantt chart for remaining tasks

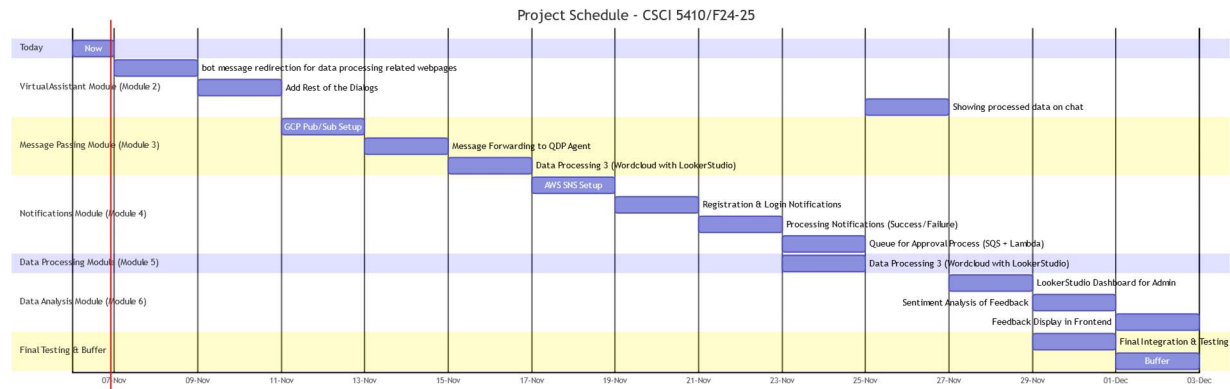


Figure 26: Gantt chart for remaining tasks

Meeting Logs

Table 2: Shows Team 4 meeting logs for sprint 1.

Meeting Date	Meeting Link	Attendees
21/10/2024	Call with Alishan and 3 others-20241021 130652-Meeting Recording.mp4	Everyone from the team joined
29/10/2024	Meeting in Serverless Group Project-20241028 231235-Meeting Recording.mp4	Everyone from the team joined
06/11/2024	Call with Serverless Group Project-20241106 214535-Meeting Recording.mp4	Everyone from the team joined

References:

1. AWS, "Getting started with user pools," Amazon Web Services, 2023. [Online]. Available: <https://docs.aws.amazon.com/cognito/latest/developerguide/getting-started-user-pools.html>. [Accessed: November 04, 2024].
2. AWS, "Class: AWS.CognitoIdentityServiceProvider," Amazon Web Services, 2023. [Online]. Available: <https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/CognitoIdentityServiceProvider.html>. [November 04, 2024].
3. AWS, "Tutorial: Using AWS Lambda with Amazon API Gateway," Amazon Web Services, 2023. [Online]. Available: <https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway-tutorial.html>. [November 04, 2024].
4. "DynamoDB Documentation," AWS [Online]. Available: <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide>. [Accessed: November 04, 2024].
5. React Documentation, "Getting Started with React," Meta, 2023. [Online]. Available: <https://reactjs.org/docs/getting-started.html>. [Accessed: November 05, 2024].
6. GitLab Documentation, "GitLab CI/CD Basics," GitLab, 2023. [Online]. Available: <https://docs.gitlab.com/ee/ci/>. [Accessed: November 05, 2024].
7. Google Cloud, "Getting Started with Cloud Build," Google, 2023. [Online]. Available: <https://cloud.google.com/cloud-build/docs>. [Accessed: November 06, 2024].
8. Google Cloud, "Deploying to Cloud Run," Google, 2023. [Online]. Available: <https://cloud.google.com/run/docs/deploying>. [Accessed: November 06, 2024].