Project Title

(Try to choose a catchy title. Max 20 words).

Summarize.it

Team Information

Team Name	**S-Class**
Team member 1(Team Lead, AI Engineer) Name: Saxena, Rishabh email: rsaxena3@uci.edu Student-ID: 72844211	Saxena, Rishabh – 72844211
Team member 2 (Frond-End Engineer) Name: Somani, Amit email: somania1@uci.edu Student-ID: 94206960	Somani, Amit– 94206960

Team member 3 (Back-end Engineer) Name: Sohni, Jash Ashwin email: jsohni@uci.edu Student-ID: 86168994	Sohni, Jash Ashwin– 86168994

Project Description

Motivation

(Describe the problem you want to solve and why it is important. Max 300 words).

In the modern age of the internet, a majority of the information is shared in textual form. Most of this information comprises of various documents such as news articles, legal documents, and medical records. To convey the main ideas of a document quickly and effectively, the most effective technique is to summarize the original document in a short paragraph. However, such a task requires a considerable investment of time and effort to be done manually.

With the recent advancements in the field of Artificial Intelligence and Natural Language Processing, it is now possible to design automated systems that are capable of summarizing an input document. This could help immensely in various fields such as summarizing news articles to save the reader's time. We find this to be an interesting, challenging, and useful problem to solve. Therefore we have decided to work on a project that aims to create a web-app platform that would allow users to generate shorter versions of input documents provided.

State of the Art / Current solution

(Describe how the problem is solved today (if it is). Max 200 words).

The task of document summarization can be tackled as two separate subproblems, namely extractive text summarization and abstractive text summarization. Extractive Text Summarization refers to the task of picking out sentences from the input document which collectively contain the most important information whereas abstractive text summarization refers to paraphrasing the whole document with completely new sentences. Since abstractive text summarization is slightly more difficult to do and still an emerging research topic, we will primarily focus on extractive text summarization techniques for this project.

For the task at hand, the state of the art research work can be found here. The current state of the art solution has a score of 43.85 for the ROUGE1 metric (which is a standard metric for automatic summarization evaluation in NLP).

Additionally, a web application called Resoomer provides similar functionality to summarize input documents and it also performs extractive techniques.

Project Goals

(Describe what are the outcomes of the project and how you will conduct a short final demo. Max 200 words).

The primary goals and deliverables of our project are the following:

1) Train a machine learning model using NLP techniques to generate text summaries. Since the text data can be in many formats, we will only focus on plain-text data, and if time permits, we will add support for more types of data formats.
2) Develop user-friendly web applications to generate text summaries and export them as .pdf, .doc documents in 1-click.
3) Expose a text summarization API allowing external users to integrate with our service.
4) Deploy the model and web application on the cloud.
5) While working on this project, we wish to gain a lot of knowledge in creating a web application and working on a Natural Language Processing Task.

How the demo will be presented?

All the features of the web application will be presented in a recorded video format.

Assumptions

(Describe the assumptions (if any) you are making to solve the problem. Max 180 words).

We will have the following assumption in our project:

1. Since we are working on an Extractive Text Summarization based solution, we will assume that the user input can be summarized well using extractive techniques. This is generally true for factual documents like New Articles, Encyclopedia entries, Legal Documents, & Medical Reports. However, such a solution might not produce good results for other pieces of subjective writing such as trying to summarize a novel. Our solution is not intended for such written text.
2. The primary use-case we would like to solve is to try to summarize news articles. Therefore, this will be our primary focus for dataset selection, training, and testing.
3. Since our primary goal is to create a Web Application and the NLP model, we will not focus on converting raw documents into textual form. This means we would allow users to only provide in textual form and will not focus on parsing .pdf/.doc and other file formats.
4. We will be designing and testing the service to have a maximum of 100 concurrent users simultaneously

SYSTEM ARCHITECTURE OVERVIEW

High-Level Diagram

(xxx. Max 500 words or 1 page).

The following 3 figures provide information on the High-Level Diagram of the various components of our system. The diagrams primarily illustrate how the user will interact with the front end application which will internally call the back end application. The back end application will then route the request to the ML model of the database according to the specific user request. Please note that the following diagrams show the storage service used as Amazon S3. We are still working on identifying the best service for our requirements and might end up using Dynamo DB instead.

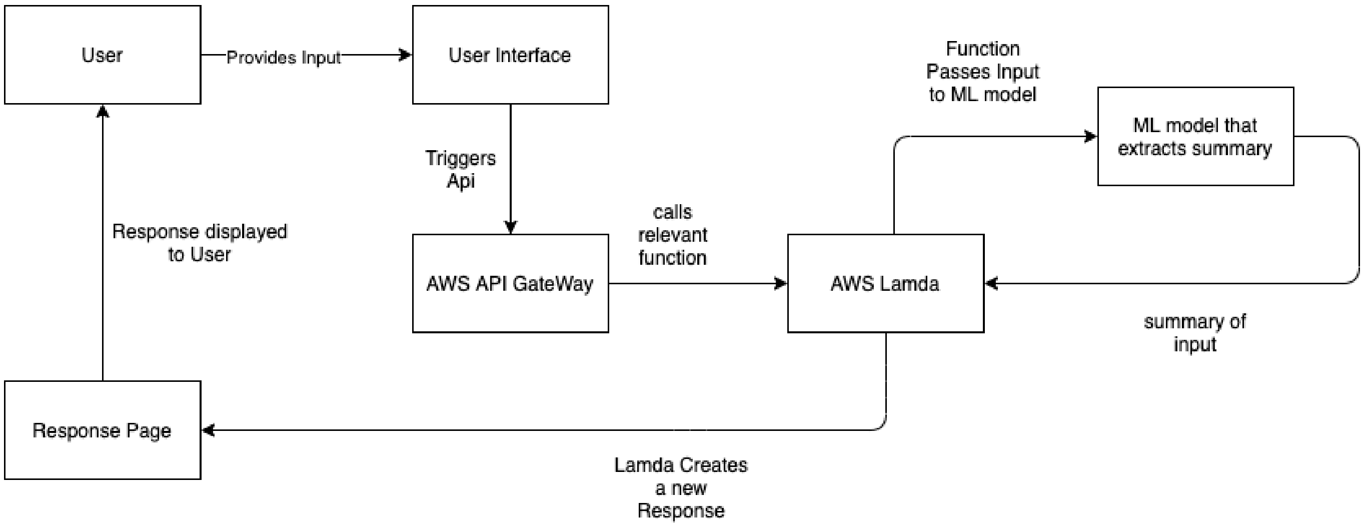

Figure 1: *This diagram illustrates the data flow of generating a summary from the user providing the input to the UI, to it being passed to the ML model via the AWS lambda backend service and then the response is relayed back to the user via the front end application.*

(Please proceed to next page for diagrams of other flows)

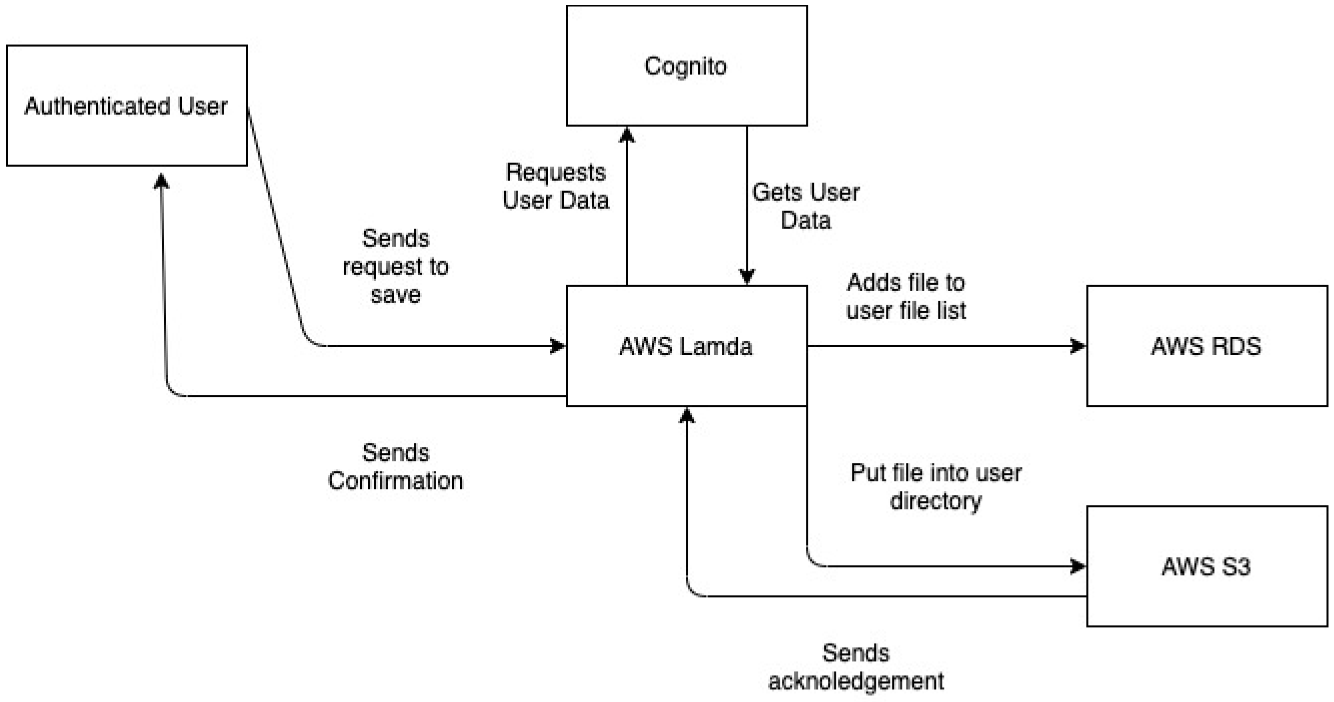

***Figure 2**: This diagram depicts the flow when a user who is already authenticated tries to initiate a request to save a generated summary with the benefit of having access to it at a later stage. The request is passed from the front to the corresponding API in AWS which can perform the store operation on Amazon S3/ Dynamo DB.*

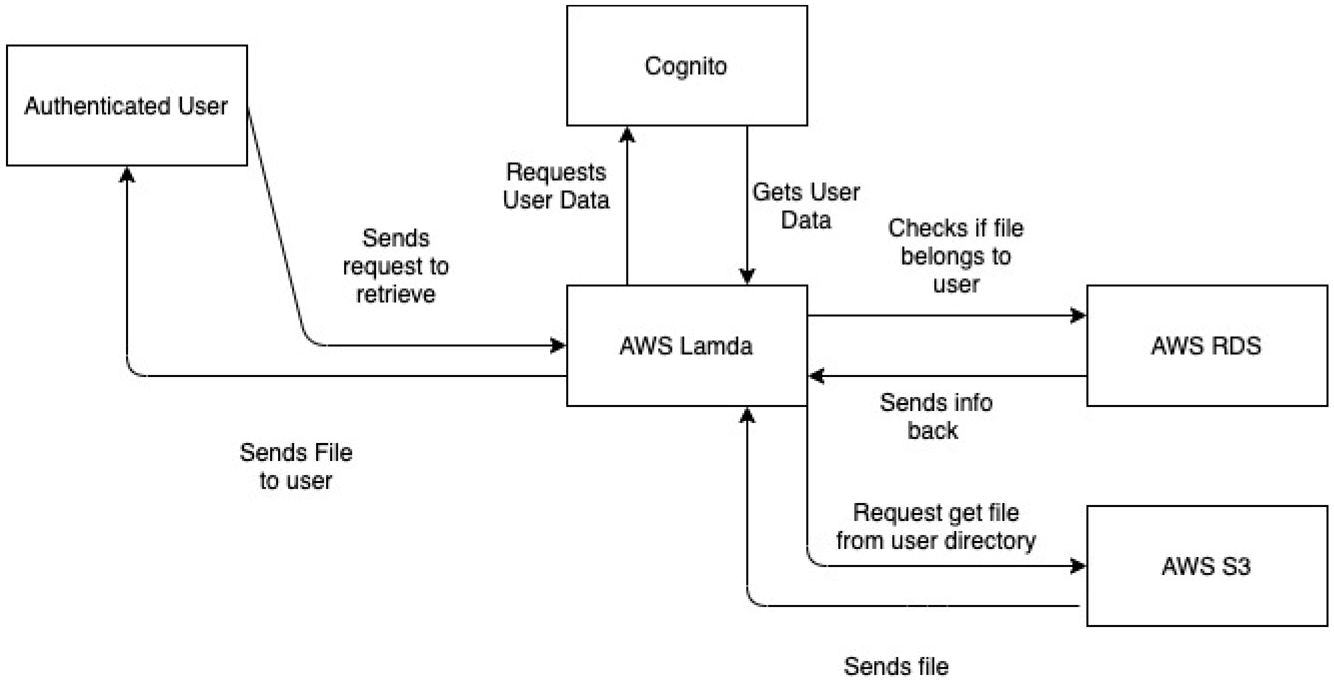

***Figure 3:** This diagram depicts the flow of retrieval of an existing saved response on user request.*

User Interaction and Design

(xxx. Max 500 words or 1 page).

In this section, we will provide some initial mockups of the front end. This will hopefully provide some insight into how the user would be able to seamlessly provide different inputs and generate corresponding summaries.

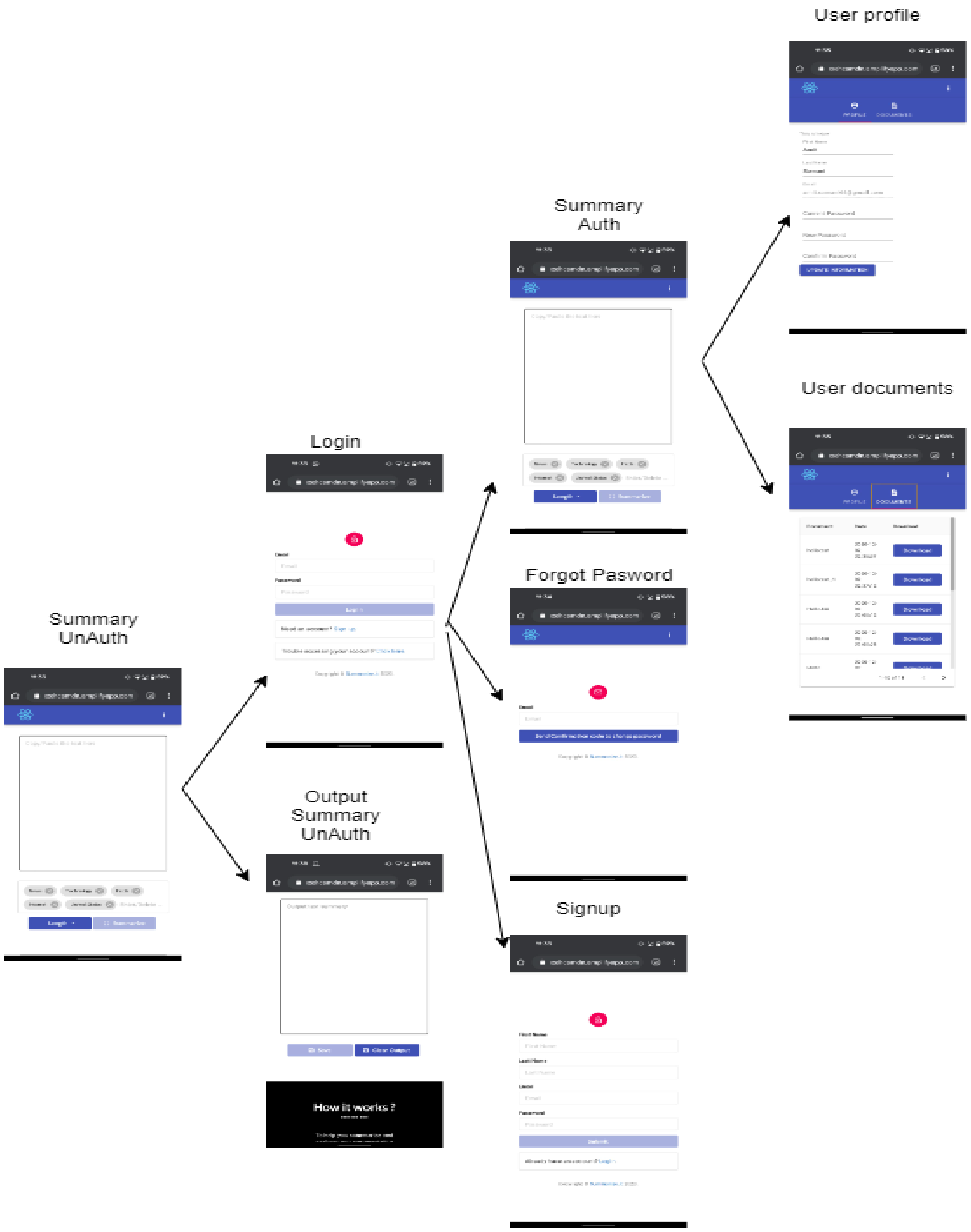

Figure 10: *Mobile Web-App UI flow*

SYSTEM MODELS

(Provide a visual design of the solution by showing contexts, sequences and behavioral sates using The Unified Modeling Language (UML). Max 1,000 words or 2 pages).

Since the various interactions between the front-end application, back-end application, and the ML model are explained in the high-level diagram, we will explain the various user flows using sequence diagrams. Since our application is not very complex from the perspective of software design and leans more on interdisciplinary fields (like NLP, Machine Learning), we don't think UML diagrams provide a lot of useful information. However, Figure 13 does include a UML diagram for the database schema that we are planning to implement for the save summary feature. Figure 14 illustrates a user flow diagram which highlights all the various user interaction in a state machine representation.

Here we will provide a few sequence diagrams for the various flows of our system:

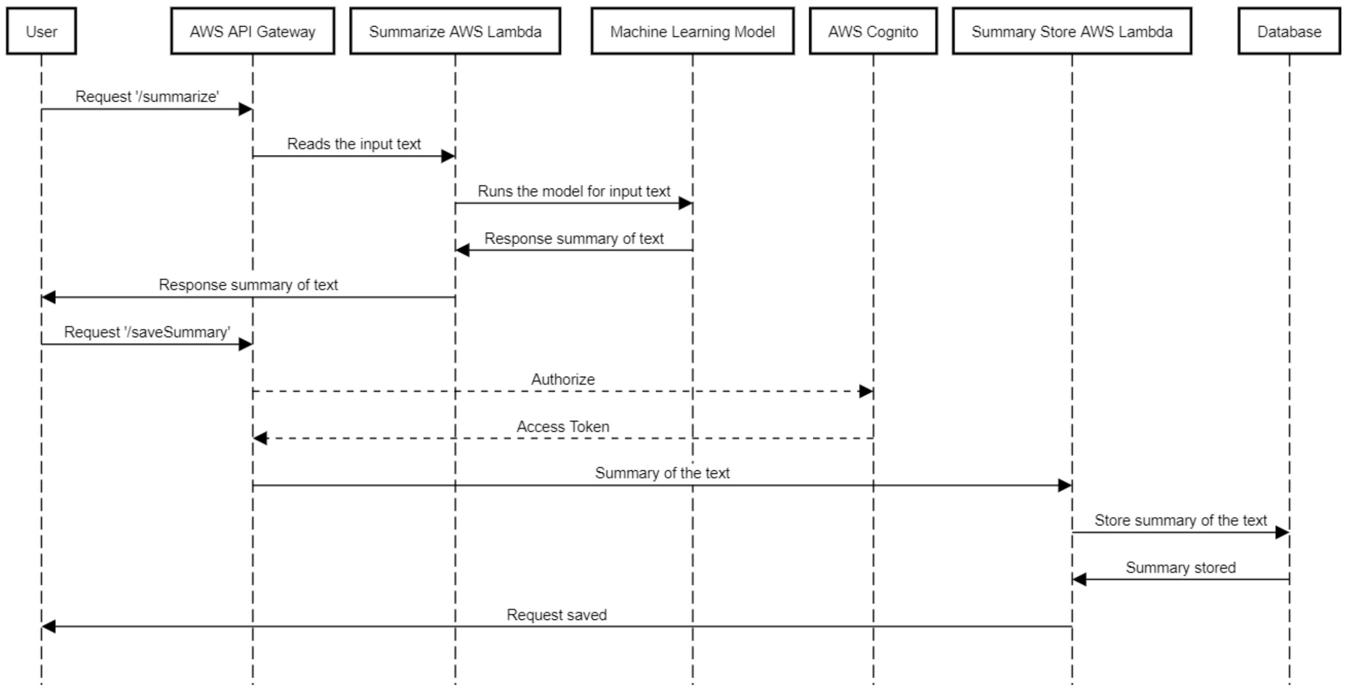

Figure 11: *Sequence Diagram of a user generating a summary of the input text*

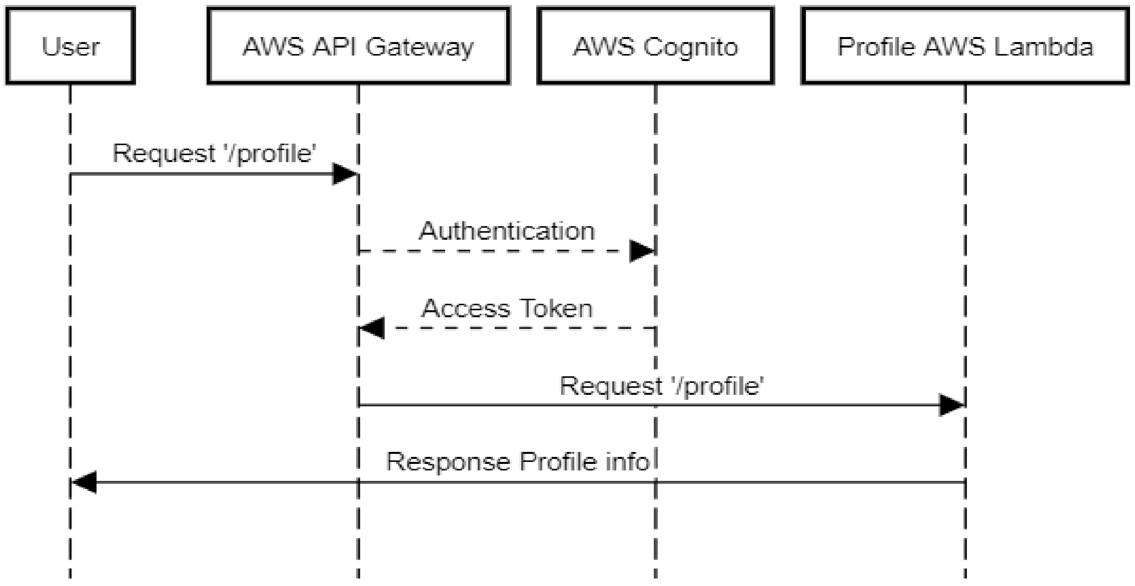

***Figure 12**: Sequence Diagram of a user accessing the summarize.it profile*

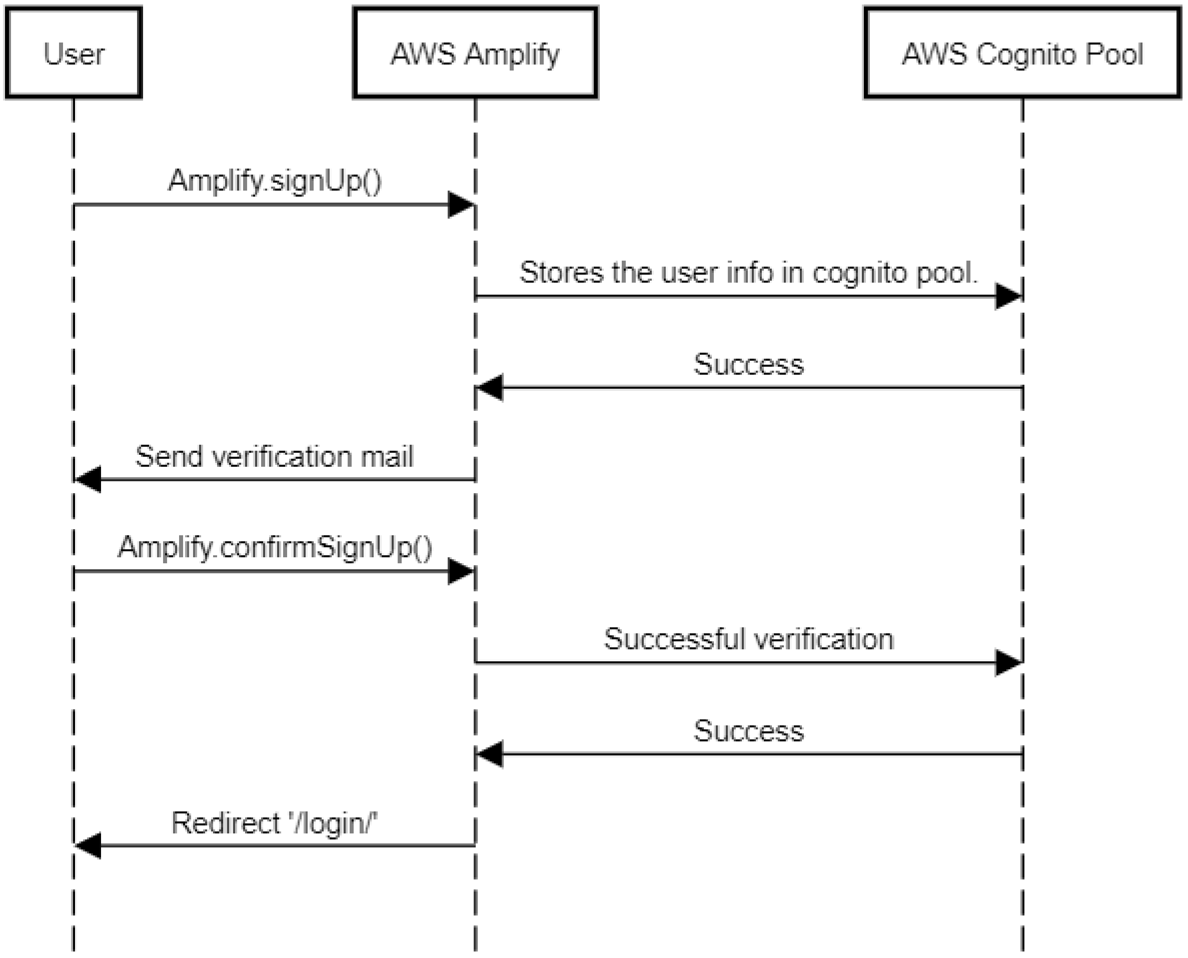

***Figure 13**: Sequence Diagram of a user signing up for the service.*

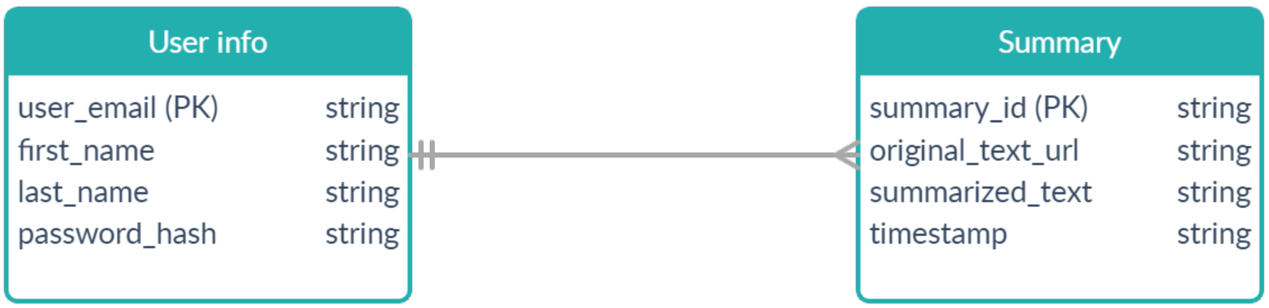

Figure 14: *Database UML diagram*

+

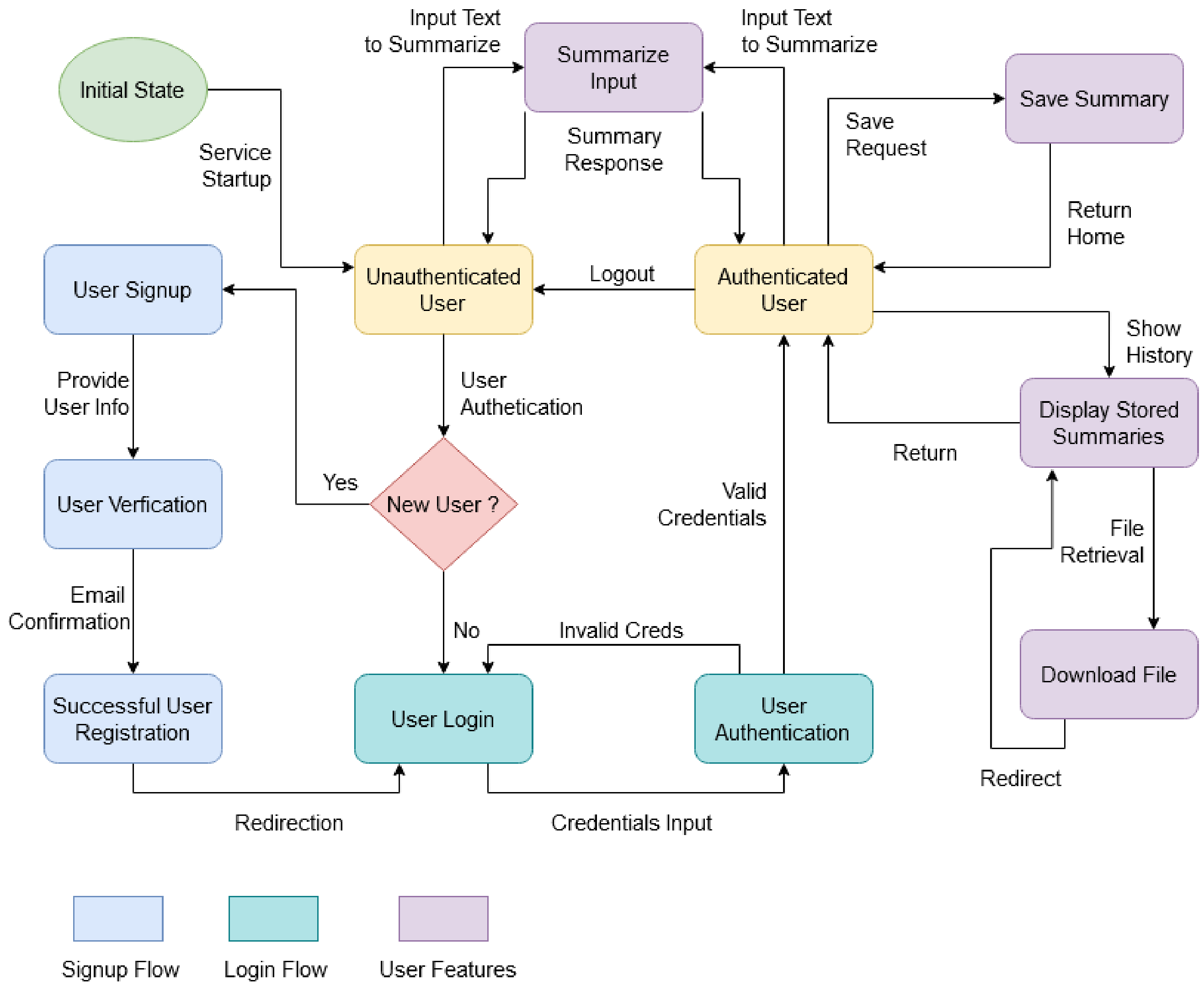

Figure 15: *Front End state flow for user actions.*

FINAL SOFTWARE TOOLS/PACKAGES UTILIZED (5 pts)

Software Tools/Packages
Front-End Development: React.js, Axios, aws-amplify, material-ui, semantic-ui, aws-sdk
Back-End Development: Python, aws-lambda, aws-apigateway, aws-sdk, aws-s3, aws-rds
Model Development: Python, matplotlib, sklearn, numpy, pandas, aws-sagemaker, aws-lambda, nltk.

PROJECT IMPLEMENTATION (20 PTS)

Tasks/Milestones Completed (10 pts)

Task Completed	Team Member
Literature review for the summarization model.	Rishabh Saxena
Data preprocessing for the summarization model	Rishabh Saxena
Model training for summarization	Rishabh Saxena
Model Evaluation for summarization	Rishabh Saxena
Model deployment using AWS Sagemaker and AWS Lambda	Rishabh Saxena
Designing and evaluating various back-end architectures for the web application	Jash Ashwin Sohni
Implementing Authentication Service using AWS Amplify	Jash Ashwin Sohni
Implementing Summarization API using AWS Lambda serverless	Jash Ashwin Sohnni
Implementing Summary Save API using AWS Lambda	Jash Ashwin Sohnni
Designing database for the storing and fetching summaries using S3 and AWS RDS	Jash Ashwin Sohnni
Designing UI for the 1-click summarization of text	Amit Somani
Designing UI flow for authentication and verification	Amit Somani

Designing User dashboard including profile and user’s document list	Amit Somani
Designing UI for providing summary text control handles like length and context	Amit Somani
Integrating UI, Backend, and Model	Team S-Class

Challenges/Roadblocks (5 pts)

(Describe the challenges that you have faced and how you solved them if that is the case. Max 300 words).

- **Implementing scalable API for summarization**: Initially, we decide to implement back-end using AWS EC2 but hit a roadblock in deciding how do we scale and manage resources for EC2 effectively as we are a 3 person on the team. To overcome this, we decided to re-design our backend stack using AWS Serverless architecture, which will take care of scalability and availability for us on-demand.
- **Integration of various services for the summarize.it:** Integrating machine learning model running on aws-sagemaker and also front-end with aws-lambda through aws-apigateway proved to be a difficult task for us as this was the first time any of us were using the aws serverless stack. We had various api authentication, cross aws account access, and CORS issues. But as we read more documentation, we became clear of the scripts that we need to write for integration and successfully integrated the services.
- **Understanding the best approach for summarization:** In the design phase of the project, we considered various solutions, and almost all of the current research approaches are based on Deep Learning. Implementation of these models requires a lot of domain expertise. Therefore, as a solution, we focused on implementing a simpler approach using extractive techniques and glove embeddings.

Tasks Not Completed (5 pts)

(Describe the tasks that you originally planned to complete but were not completed. If all tasks were completed, state so. Max 250 words).

We have completed all the critical tasks for summarize.it as initially planned. However, one task related to testing wasn’t completed which has been listed below.

Task	Reason
Running performance tests with 100 concurrent users.	It took us more time than expected to complete the integration stage of our project which resulted in lesser time for testing. Therefore we focused on functional testing and deprioritized this task as scalability is related to AWS infrastructure.

Week points / Future work (5 pts)

(Mention at least two points of your project that have room for improvement. These points can be additions to the existing project requirements or improvement of the current implementation. Max 200 words).

Below are a few features that we would love to implement if given more time to improve the product and user experience.

Task	**Reason**
Allowing users to give input text in .pdf, .doc, or similar format instead of just raw text.	Was not part of the milestones for MVP.
Developing a web browser extension to summarize selected text on any website without having to visit the website. (e.g. Grammarly Chrome Extension)	Was not part of the milestones for MVP.
Allowing users to preview saved summaries before downloading them from the cloud.	Was not part of the milestones for MVP.

FINAL DEMO VIDEO (20 pts)

Please include a link to a demo video (5 min max) showing the functionality of your project. A Google drive link is preferred.

https://drive.google.com/file/d/11RF3rve2OBk594qpUvBLkrYaHHJPnEAV/view?usp=sharing

SOURCE CODE (15 pts)

Please include a link to your project repository (such as UCI GitHub).

Github: https://github.com/rsaxena07/summarize.it