Rayce Ramsay (1009734888)
Ali Shabani (1008838652)
Terry Tian (1007663468)
Grant Hamblin (1009860447)

CSC490 Assignment A3 - Deleting Prod

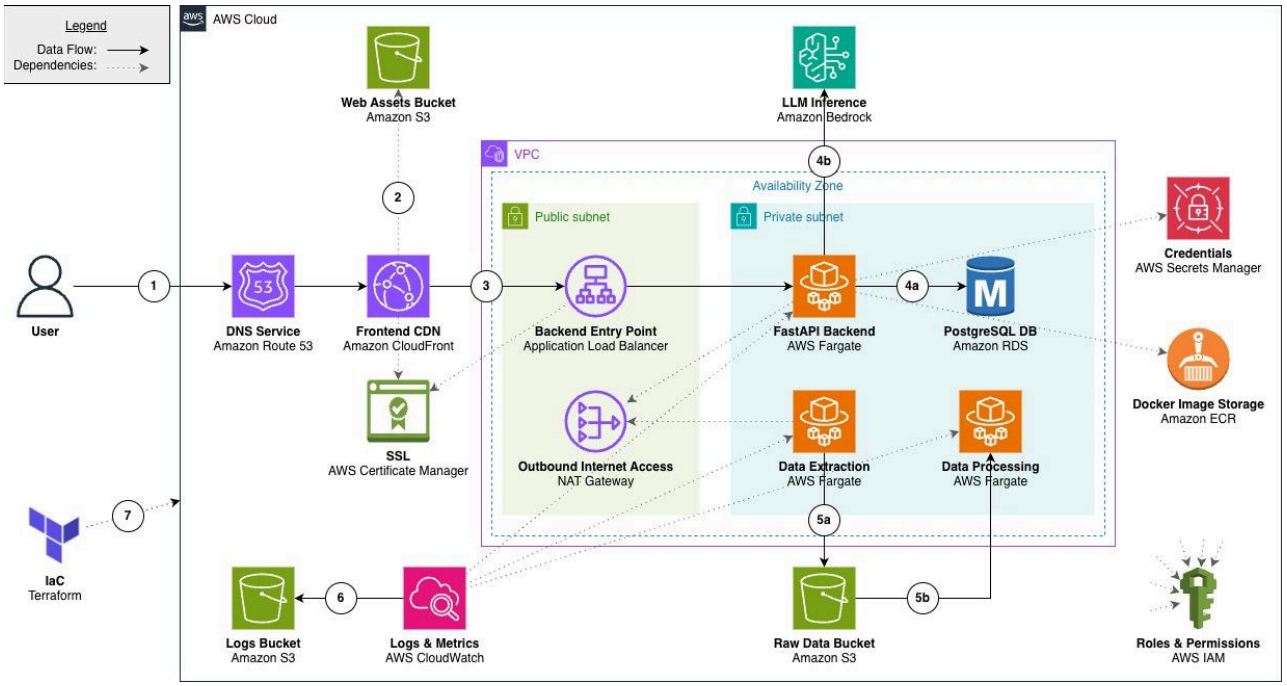
1 Part One: Diagramming Your Infrastructure (20 marks)

Create a detailed diagram of your project's current infrastructure. Your diagram should include:

- Application services
- Databases
- Data pipelines
- Other relevant components

If certain components are planned but not yet implemented, include them and clearly mark them as future state. Highlight any differences between your development and production environments. You will be awarded marks for the clarity and completeness of the diagram

AWS Infrastructure Diagram:



Application Flow (following the numbers in the previous diagram):

1. User accesses https://aidoctors.com. Route 53 resolves the domain and routes traffic to the CloudFront distribution.
2. CloudFront delivers static Next.js assets from the web-assets S3 bucket, and the browser loads the UI (HTML, CSS, JS).
3. The user submits input (patient characteristics, current medications, conditions, and a new drug). The browser sends an HTTPS request to https://api.aidoctors.com. routed by Route 53 and the Application Load Balancer (ALB) to a containerized FastAPI backend running on AWS Fargate.
4. The backend:
    a. Queries the RDS PostgreSQL database for similar patients and existing drug-drug interaction (DDI) data.
    b. Sends the combined input, patient data, and DDI data to Amazon Bedrock for LLM inference, generating the top-k (k TBD) most severe DDI alerts and returning the results to the user (via ALB → CloudFront → browser).
5. Data ingestion and preparation are handled by two components:
    a. A containerized Fargate task for data extraction, manually triggered to fetch external data and write raw CSVs to the raw-data S3 bucket.
    b. A containerized Fargate task for data processing/loading, manually triggered to load raw CSVs, then clean, validate, normalize, and insert them into RDS.
6. CloudWatch captures logs from the backend and data tasks and stores them in the logs S3 bucket for monitoring and diagnostics.
7. Terraform serves as the Infrastructure-as-Code (IaC) framework to provision and manage all AWS resources, maintaining distinct dev and prod environments. GitHub Actions handle CI/CD, automatically running workflows on code merges to apply Terraform updates when infrastructure changes are detected.

Prod vs Dev Environment Differences:

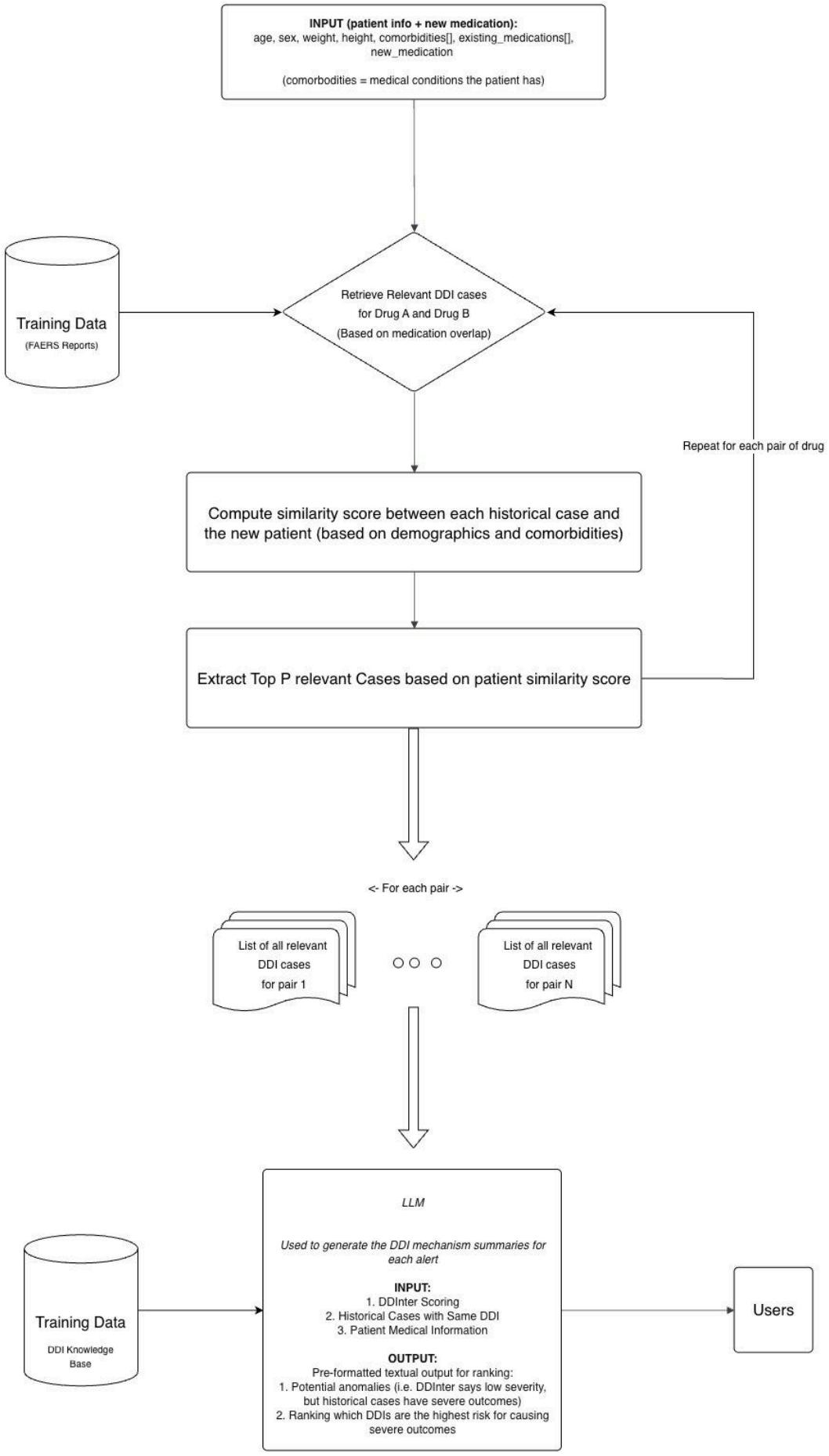
While we don't anticipate our application to receive any amount of traffic that would require significant differences between a production and development environment, we will be implementing the following for learning purpose

- In the production env we will:
    - Improve reliability by increasing the number of database replicas
    - Add autoscaling to ECS tasks and blue/green deployment strategy
    - Exhaustive logging and longer retention times (logs and versioning)
- The primary focus of development is to achieve the minimum viable replicated version of production at the lowest cost.

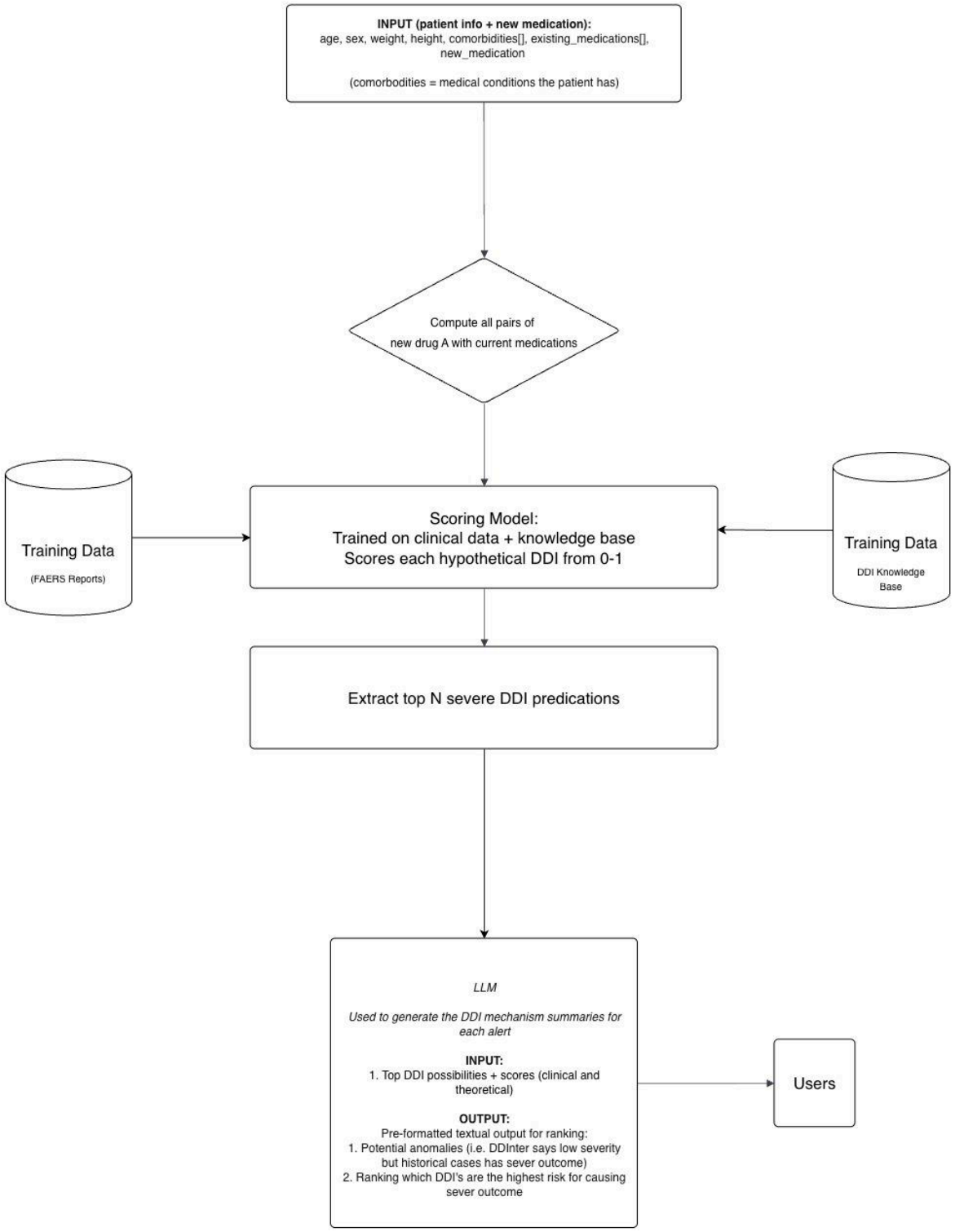
Future Infrastructure Implementations:

- Enable HTTPS with an ACM certificate (need to buy a domain first)
- Implement ECS auto-scaling based on CPU/memory
- Add RDS read replicas for read-heavy workloads
- Look at implementing AWS WAF for security
- Add CloudFront CDN for static assets
- Separate dev/prod environments
- Implement AWS Secrets Manager for application secrets
- Create ECR Images for the application and data pipeline code and verify deployment to Fargate instances.

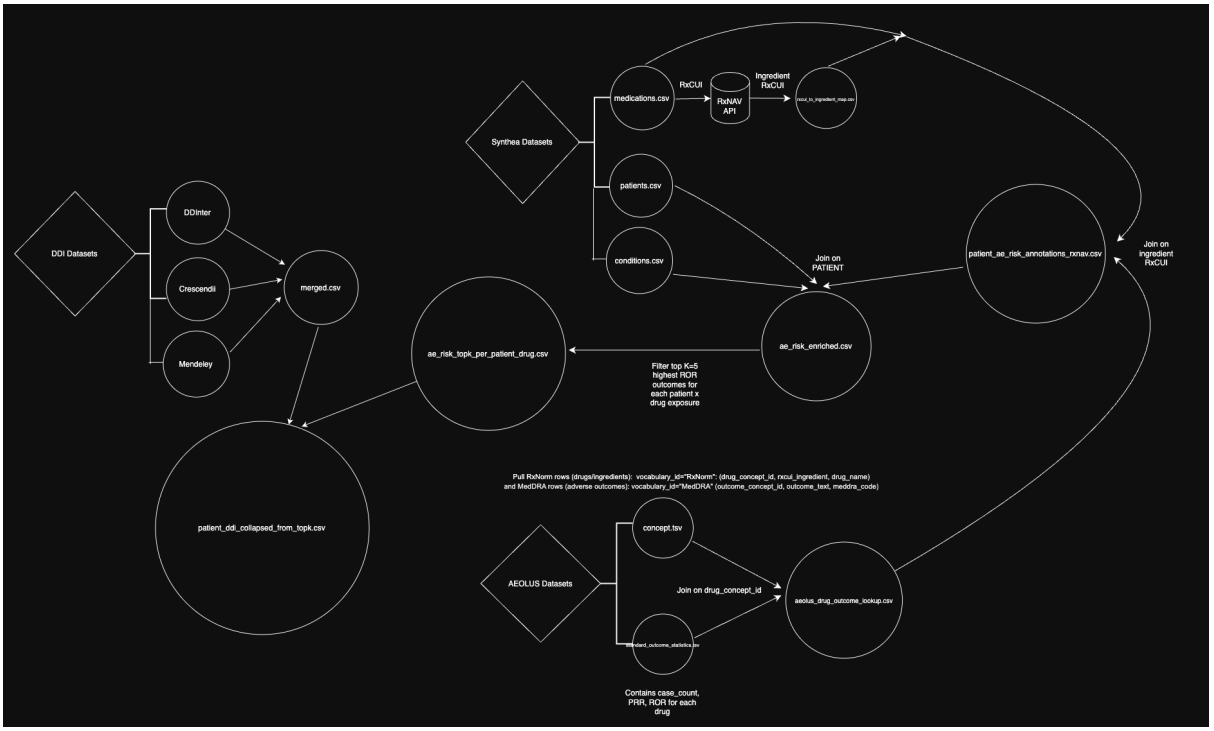
Data Flow Diagram (Iteration 1 - to be implemented):



Data Flow Diagram (Iteration 2 - future consideration):



Data Processing Diagram:



2 Part Two: Infrastructure as Code Implementation (30 marks)

Implement your infrastructure using code. You may choose any IaC framework, but your deployments should be straightforward and maintainable. For infrastructure components without existing modules:

- Create your own modules when possible (e.g., creating a Terraform module instead of using scripts)
- If using scripts, provide a clear justification for this choice

Marking criteria:

- Code clarity and organization
- Infrastructure completeness
- Alignment with the diagram from Part One
- Implementation of multiple environments

To view our Terraform code, see the “terraform-bootstrap” and “terraform” folders in our GitHub repo at https://github.com/UofT-CSC490-F2025/AIDoctors/tree/main. Please read the README files for more information.

3 Part Three: Disaster Recovery Demonstration (50 marks)

Disaster has struck! Record a screen capture of your team deleting your production environment and your IaaC to restore it. Marks will be given on the completeness of the deletion restoration and clarity of the process/code. Your restoration should include:

- Application services
- Database systems and their data
- Configuration settings
- Access controls and security settings
- Verification of system functionality

Click this Google Drive link to access our screen capture for the destruction and restoration of the production environment. Note that deployment of ECR images for data processing scripts and application code will be included in A4. The database population scripts exist and have been tested locally, but are in the process of being integrated into the cloud environment.

4 Submission Instructions

Submit a pull request to the course GitHub repo with your assignment in a folder named a3 with your a3.pdf on a branch called a3, and include your team members' names on the first page with student IDs.