**BloodBridge: Optimizing Lifesaving Resources Using AWS Services**

### A PROJECT REPORT

#### Submitted by

**1.RAMASUBRAMANIYAN K - 412622149021**

**2. SINDHUJA S - 412622149027**

**3.INDHUJA S - 412622149013**

**4. DHANASRI - 412622149006**

#### in partial fulfillment for the award of the degree of

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)**

**SRI VENKATESWARAA COLLEGE OF TECHNOLOGY**

## ANNA UNIVERSITY : CHENNAI 600 025

**NOVEMBER 2024**

## BONAFIDE CERTIFICATE

Certified that this project report “**BloodBridge: Optimizing Lifesaving**

**Resources Using AWS Services”** is the bonafide work of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_who carried out the project work under my supervision.

###### SIGNATURE SIGNATURE

###### PROJECT COORDINATOR NM SPOC

###### 

**INTERNAL SIGNATURE EXTERNAL SIGNATURE**

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGENO.**

**ABSTRACT 3**

**LIST OF TABLE**

**LIST OF FIGURES**

**LIST OF SYMBOLS**

**1. INTRODUCTION 6**

1.1 BACKGROUND

1.2 OBJECTIVE

1.4 SCOPE

**2. LITERATURE REVIEW 7**

2.1 EXISTING BLOOD DONATION

MANAGEMENT SYSTEMS

2.2 AWS and CLOUD COMPUTING IN

HEALTHCARE

2.3 RELEVANT RESEARCH

**3. SYSTEM ARCHITECTURE AND 8-9**

**TECHNOLOGIES USED**

**4. PROJECT FLOW AND FUNCTIONALITIES 10-11**

**5. IMPLEMENTATION DETAILS 12-15**

**6. TESTING AND OPTIMIZATION 16-18**

**7. CONCLUSION 19**

**REFERENCES**

**Abstract**

**BloodBridge: Optimizing Lifesaving Resources Using AWS Services**

This project, **BloodBridge**, presents a web-based application to optimize blood donation and distribution using cloud technologies. It aims to create a real-time, responsive system where hospitals, blood banks, and donors can seamlessly interact to meet urgent blood needs efficiently. BloodBridge utilizes Amazon Web Services (AWS) for hosting, storage, and database management, including EC2 and RDS, to provide a robust and scalable infrastructure. The system's primary features include emergency blood requests, donor management, and inventory tracking, with user-friendly interfaces for hospital administrators, blood bank managers, and regular donors. The BloodBridge project demonstrates how AWS can enable a reliable and high-performance platform that meets critical health service needs.

1. **List of Tables**

|  |
| --- |
| * BloodBridge System Architecture Components |
| * Functional Module Descriptions |
| * AWS EC2 Configuration Settings |
| * AWS RDS Configuration Details |
| * Summary of Testing Scenarios and Outcomes |

1. **List of Figures**

|  |
| --- |
| * System Architecture Diagram |
| * AWS EC2 and RDS Connectivity Overview |
| * BloodBridge User Dashboard Interface |
| * BloodBridge EC2 Setup Workflow |
| * Sample User Registration Screen |

1. **List of Symbols, Abbreviations, and Nomenclature**

|  |
| --- |
| * **AWS**: Amazon Web Services |
| * **EC2**: Elastic Compute Cloud |
| * **RDS**: Relational Database Service |
| * **HTTP**: Hypertext Transfer Protocol |
| * **HTTPS**: Secure Hypertext Transfer Protocol |

**Chapter 1: Introduction**

**1.1Background**  
Blood donation plays an essential role in healthcare, supporting treatments for accidents, surgeries, and chronic illnesses. However, logistical challenges in matching donors and recipients, especially for rare blood types, create inefficiencies and delays that can be life-threatening. Traditional systems often rely on static databases and manual processes, which can slow down response times and make it harder for hospitals to meet urgent needs.

**1.2Objective**  
The primary goal of the BloodBridge project is to create a cloud-enabled system that integrates real-time blood request management, donor tracking, and inventory monitoring. The platform leverages AWS services for scalable deployment and efficient data management, ensuring that blood banks and hospitals can quickly locate and connect with suitable donors when emergencies arise.

**1.3Scope**  
BloodBridge is designed to serve hospitals, blood banks, and individual donors. Key functionalities include a dashboard for tracking requests, user management for donors, real-time inventory updates by blood banks, and notifications for critical needs. AWS cloud services, such as EC2 for application hosting and RDS for database management, support the system’s backend infrastructure, ensuring it remains reliable under varying loads.

**Chapter 2: Literature Review**

**2.1Existing Blood Donation Management Systems**  
Blood donation management systems have historically faced challenges in real-time responsiveness and scalability. Traditional systems involve manual processes for tracking and fulfilling blood requests, often resulting in inefficiencies and delays. Some existing digital solutions provide static databases or basic matching systems, but few incorporate real-time notification or cloud capabilities to scale with demand. This gap underscores the need for platforms like BloodBridge, which uses cloud technologies to deliver a more efficient and responsive service.

**2.2 AWS and Cloud Computing in Healthcare**  
Cloud computing has increasingly been applied to healthcare, particularly for data storage, analysis, and secure access. AWS provides a scalable infrastructure that meets the demanding requirements of healthcare applications, such as quick response times, data reliability, and ease of integration with other systems. Services like EC2 and RDS allow healthcare applications to handle high user volumes and intensive data processing without compromising performance. Additionally, AWS compliance with healthcare regulations like HIPAA makes it a suitable choice for applications involving sensitive data.

**2.3 Relevant Research**  
Studies have shown the effectiveness of cloud-based applications in improving the efficiency of healthcare services, from patient management to resource allocation. Research on using cloud infrastructure to support real-time communication and high availability further validates BloodBridge's approach of leveraging AWS to meet the operational needs of a dynamic blood donation system. By integrating best practices in cloud computing and data security, BloodBridge aims to establish a model that other healthcare applicationemulate.

**Chapter 3: System Architecture and Technologies Used**

**3.1 Overview of System Architecture**  
BloodBridge's architecture is built on Amazon Web Services (AWS) to leverage scalable and reliable cloud infrastructure. The system is divided into three main layers: the **Presentation Layer**, the **Application Layer**, and the **Database Layer**. This modular design facilitates efficient data flow and ensures each component operates independently, enhancing maintainability and scalability.

**3.2 AWS EC2 (Elastic Compute Cloud)**  
AWS EC2 provides a secure, resizable compute capacity in the cloud, which BloodBridge uses to host its web application. We selected the **t2.micro instance** (for development) due to its low-cost and high performance, suitable for handling moderate traffic in the application’s testing phase. Key features include:

* **Virtual Servers**: For application deployment and testing.
* **Security Groups**: Configured to restrict access to HTTP, HTTPS, and MySQL traffic as per security best practices.
* **Elastic IPs**: For consistent IP address assignment, simplifying DNS setup.

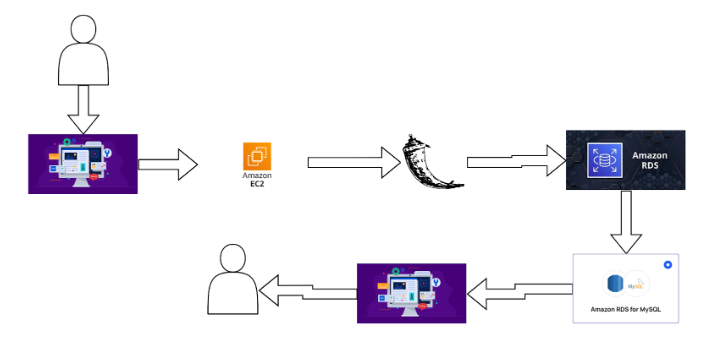
**3.3 AWS RDS (Relational Database Service)**  
AWS RDS hosts the **MySQL database** used to store critical data, including user profiles, blood request records, and inventory updates. AWS RDS ensures automatic backups, high availability, and security, which are vital for healthcare applications handling sensitive data. Configuration includes:

* **Instance Type**: db.t2.micro for cost-effectiveness.
* **Storage**: Scalable storage setup to handle increasing data demands as the application scales.
* **Networking**: Configured with VPC and subnets to ensure database security and limited access.

**3.4 Technology Stack**  
BloodBridge uses a combination of **Flask** (Python) for the backend and **HTML/CSS/JavaScript** for the frontend. This stack enables rapid development and an intuitive user interface. Flask's lightweight framework is compatible with cloud environments like AWS, making it an optimal choice for deploying a scalable web application.

**3.5 System Architecture Diagram**

The architecture diagram (placeholder for image) displays interactions between users, the EC2 server hosting the application, and the RDS database, highlighting the data flow and real-time updates. The architecture emphasizes secure communication between the components, with encrypted data transactions over HTTPS.



**Chapter 4: Project Flow and Functionalities**

**4.1 Emergency Blood Requests**

In emergency cases, hospital administrators can quickly log in to BloodBridge and submit urgent blood requests. This feature prioritizes requests by urgency and notifies relevant donors within the vicinity, significantly reducing response times. Once a request is fulfilled, the status is updated automatically to prevent redundant notifications.

**Scenario 1: Emergency Request Submission**

Sarah, a hospital administrator, logs in and submits a high-priority request. The system automatically pings nearby registered donors who match the required blood type, enabling Sarah to find a match faster.

**4.2 Donor Management**

Regular donors can manage their profiles, view upcoming eligibility dates, and receive notifications of nearby blood drives. This feature supports consistent blood availability by encouraging regular donations and making it convenient for donors to participate.

**Scenario 2: Regular Donor Interaction**

John, a regular donor, receives a reminder on his dashboard about his eligibility to donate and notices a local blood drive event. He schedules his next donation, thereby contributing to a steady blood supply.

**4.3 Blood Bank Inventory Updates**

Blood bank managers have access to inventory updates to record incoming donations and current stock. The system’s dashboard reflects the availability of each blood type, assisting in prioritizing requests for low-stock types.

**Scenario 3: Real-Time Inventory Management**

Lisa, a blood bank manager, logs in to update the blood inventory. This update is instantly visible to hospital staff, ensuring they have real-time insights into available blood types, facilitating faster response for critical needs.

**4.4 Functional Module Descriptions**

| **Module** | **Description** |
| --- | --- |
| User Registration | Allows new users to create accounts and provide essential details. |
| User Login | Authenticates users and grants access to their respective dashboards. |
| Blood Request | Allows submission and tracking of blood requests, with priority levels. |
| Inventory Management | Enables blood bank staff to update blood stock levels. |

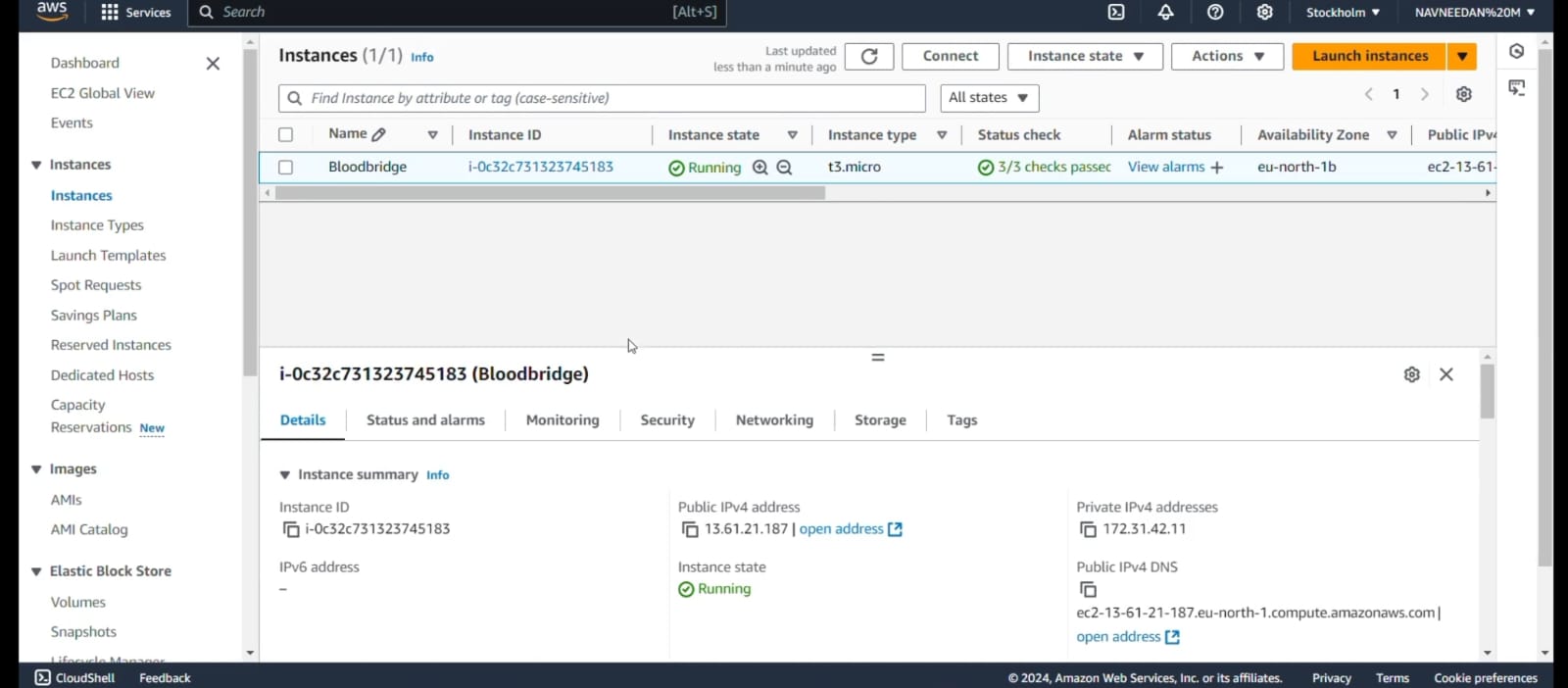
**Chapter 5: Implementation Details**

**5.1 AWS EC2 Configuration for Application Hosting**

1. **EC2 Instance Launch**: Created through the AWS console with an Amazon Linux 2 AMI, which supports the application’s dependencies.
2. **Security Group Setup**: Configured to permit HTTP/HTTPS access for web traffic and SSH for administrative access. The MySQL port is also opened but limited to secure interactions between EC2 and RDS.
3. **Elastic IP Assignment**: Ensures a stable IP address for public access.

| **Setting** | **Configuration Details** |
| --- | --- |
| Instance Type | t2.micro |
| AMI | Amazon Linux 2 |
| Security Group Rules | HTTP (80), HTTPS (443), MySQL (3306) |
|  |  |

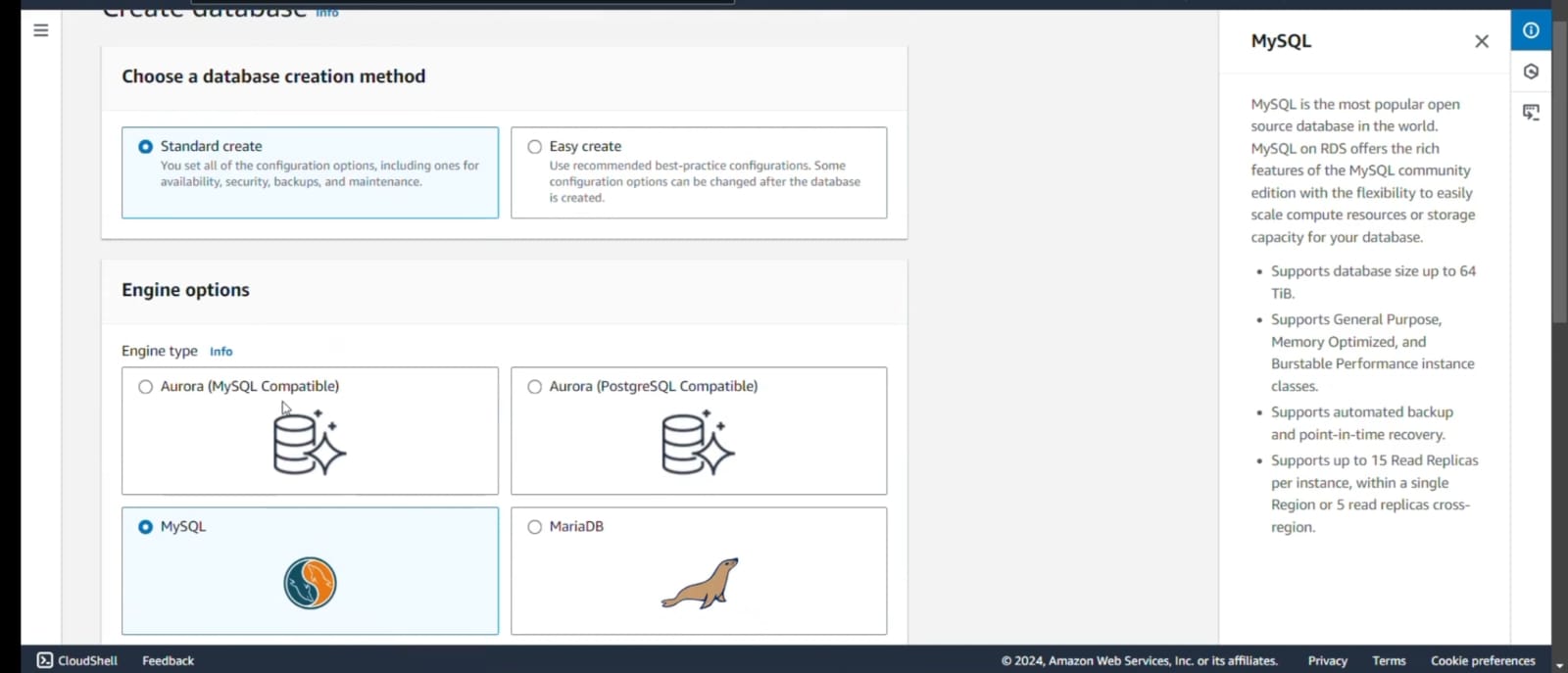
* EC2 Setup Workflow (placeholder for image) outlines the steps for launching, configuring, and managing the EC2 instance.



**5.2 AWS RDS Configuration for MySQL Database**

1. **Database Creation**: Configured through RDS with MySQL engine, appropriate security settings, and an initial database name (bloodbank).
2. **Security Group and VPC**: Ensured compatibility with EC2 by placing both in the same VPC, restricting database access to only the application server.

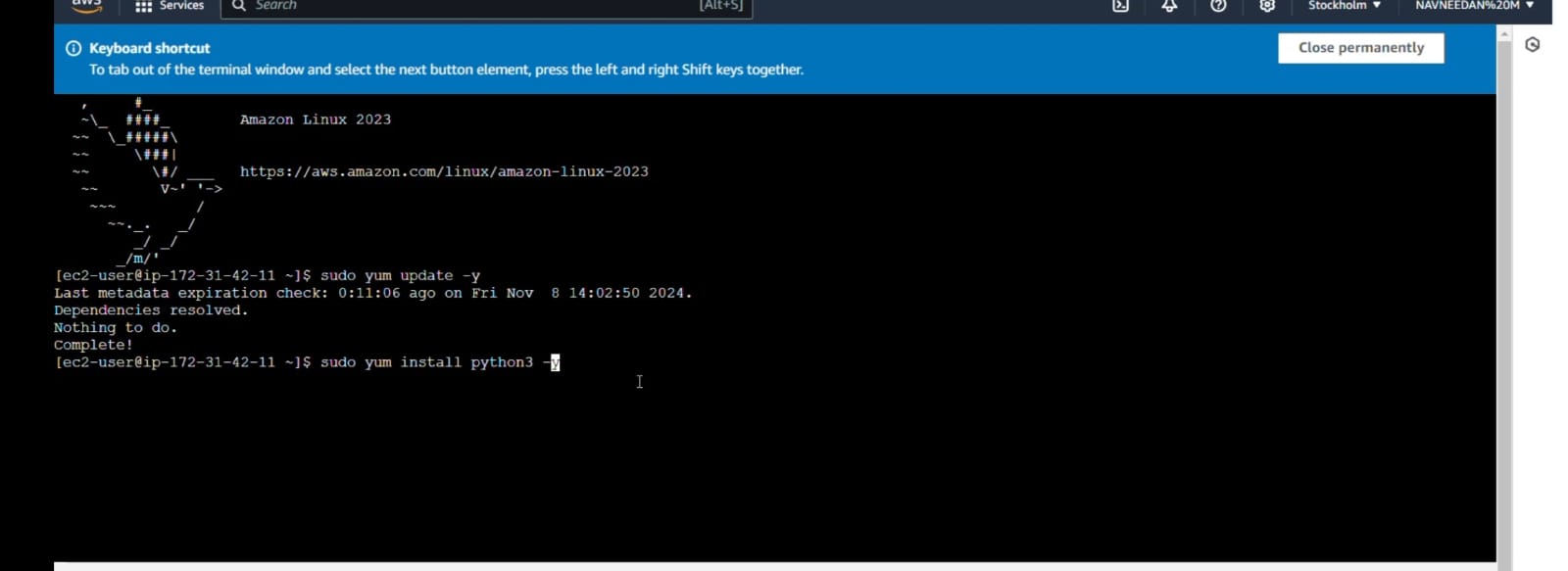
| **RDS Configuration** | **Details** |
| --- | --- |
| Engine | MySQL |
| Instance Type | db.t2.micro |
| Security Settings | Private subnets with restricted access |
|  |  |

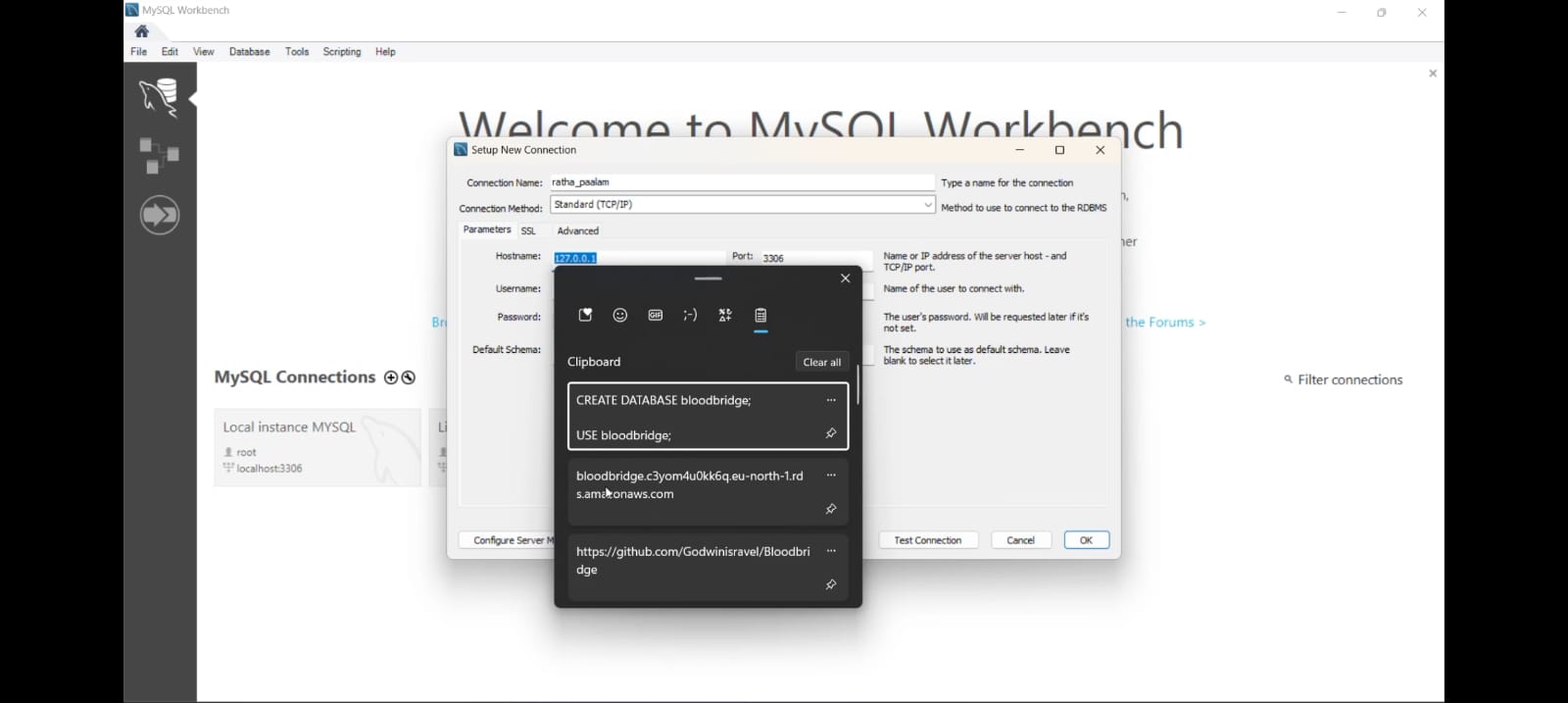
****

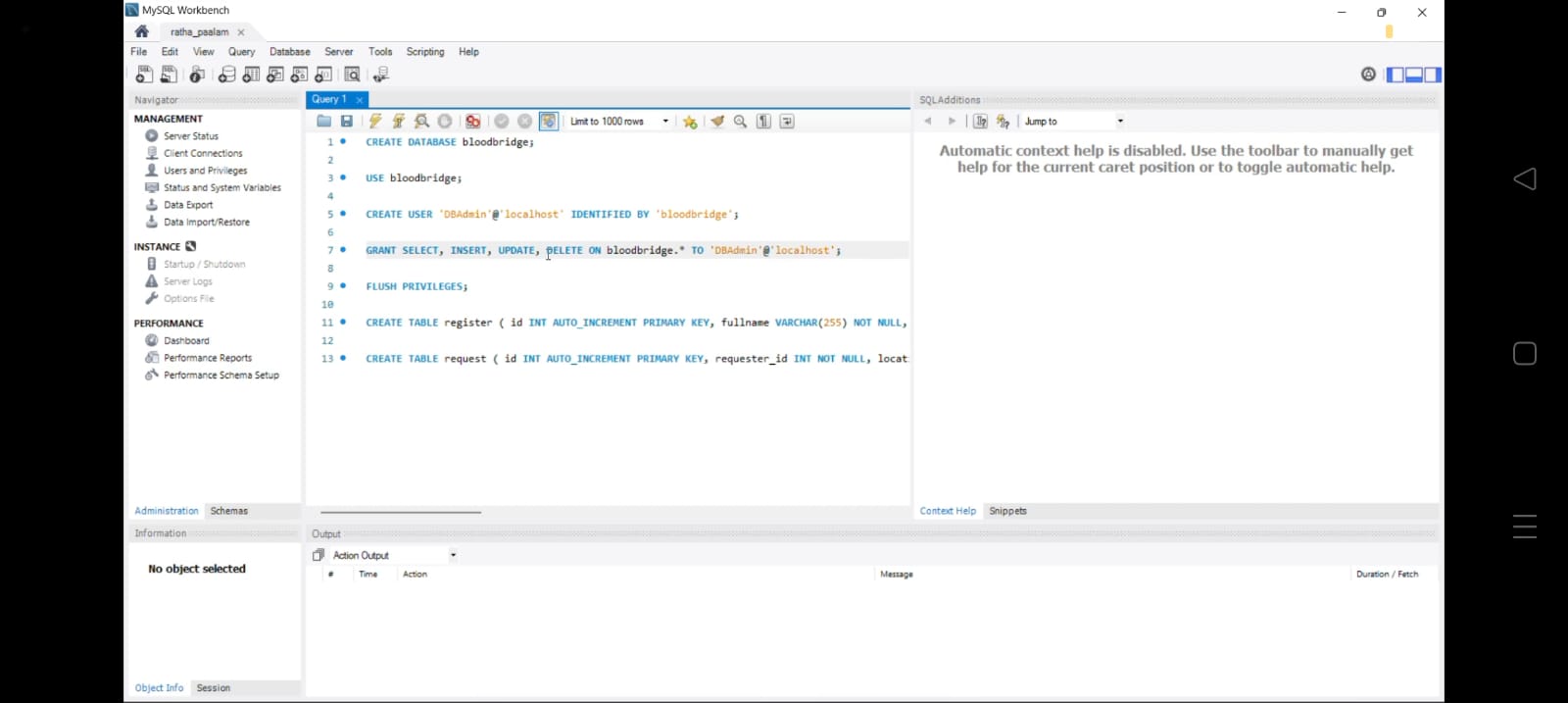
**5.3 Front-End and Back-End Integration**  
The front-end, developed in HTML/CSS/JavaScript, is connected to the Flask-based backend. Flask manages routing and data handling between the user interface and MySQL database, where interactions (e.g., blood request submissions, inventory updates) are recorded and displayed in real-time.

**5.4 Database Design**  
Tables include:

* **Users**: Stores donor and admin information.
* **Blood Requests**: Tracks requests, status, and priority.
* **Inventory**: Maintains blood type quantities for quick updates by blood banks.



****

****

**Chapter 6: Testing and Optimization**

**6.1 Functional Testing**  
Each module, from user registration to inventory updates, underwent rigorous testing to ensure accurate functionality:

* **Registration and Login**: Tested for valid and invalid input cases.
* **Blood Requests**: Verified data submission, retrieval, and update accuracy.

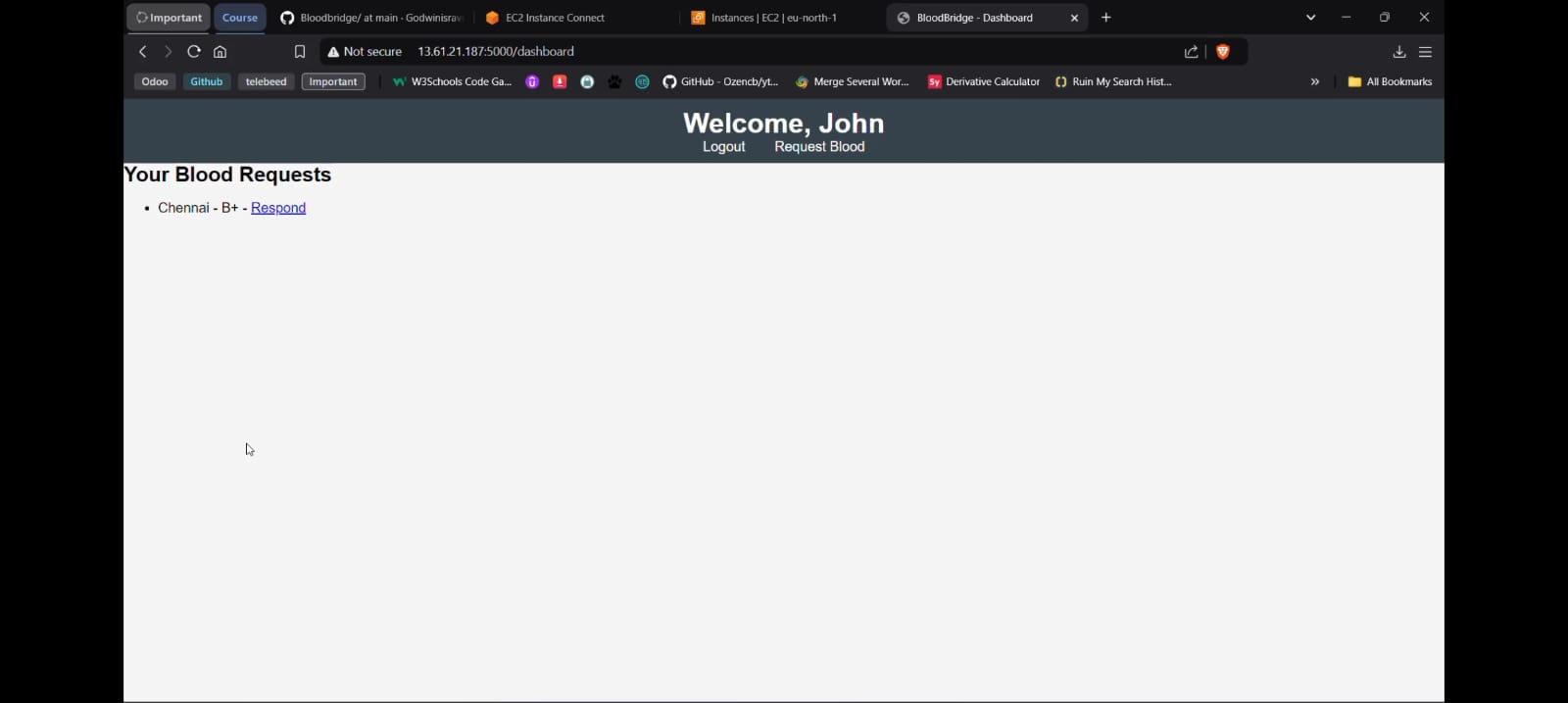
**6.2 Database Optimization**  
Optimization techniques, such as indexing frequently queried fields (e.g., user IDs, blood types), improved database response times. Additionally, reducing redundant queries minimized server load.

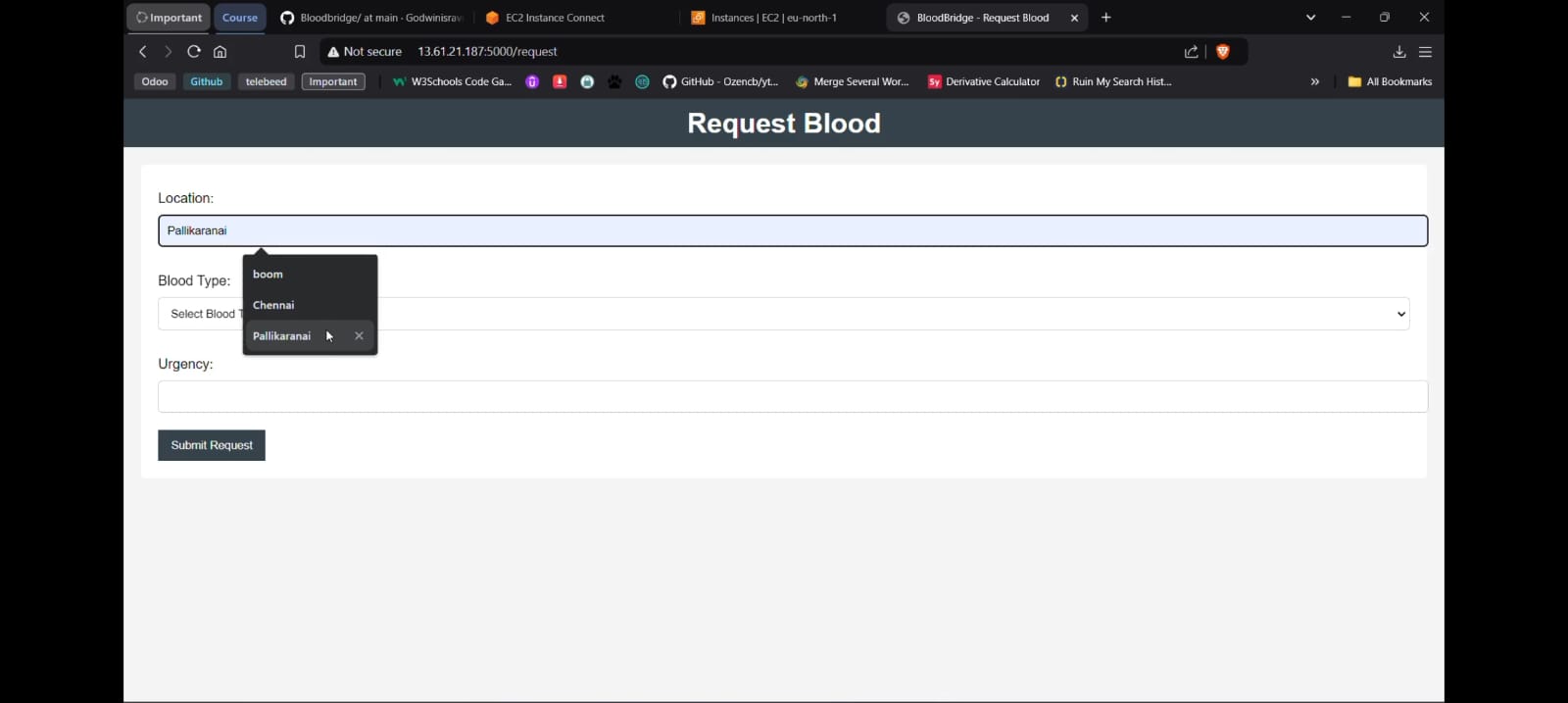
**6.3 Load Testing**  
Simulated high user volumes to evaluate EC2 and RDS scalability. Results indicated stable performance under expected load, and additional load balancers may be added as demand grows.

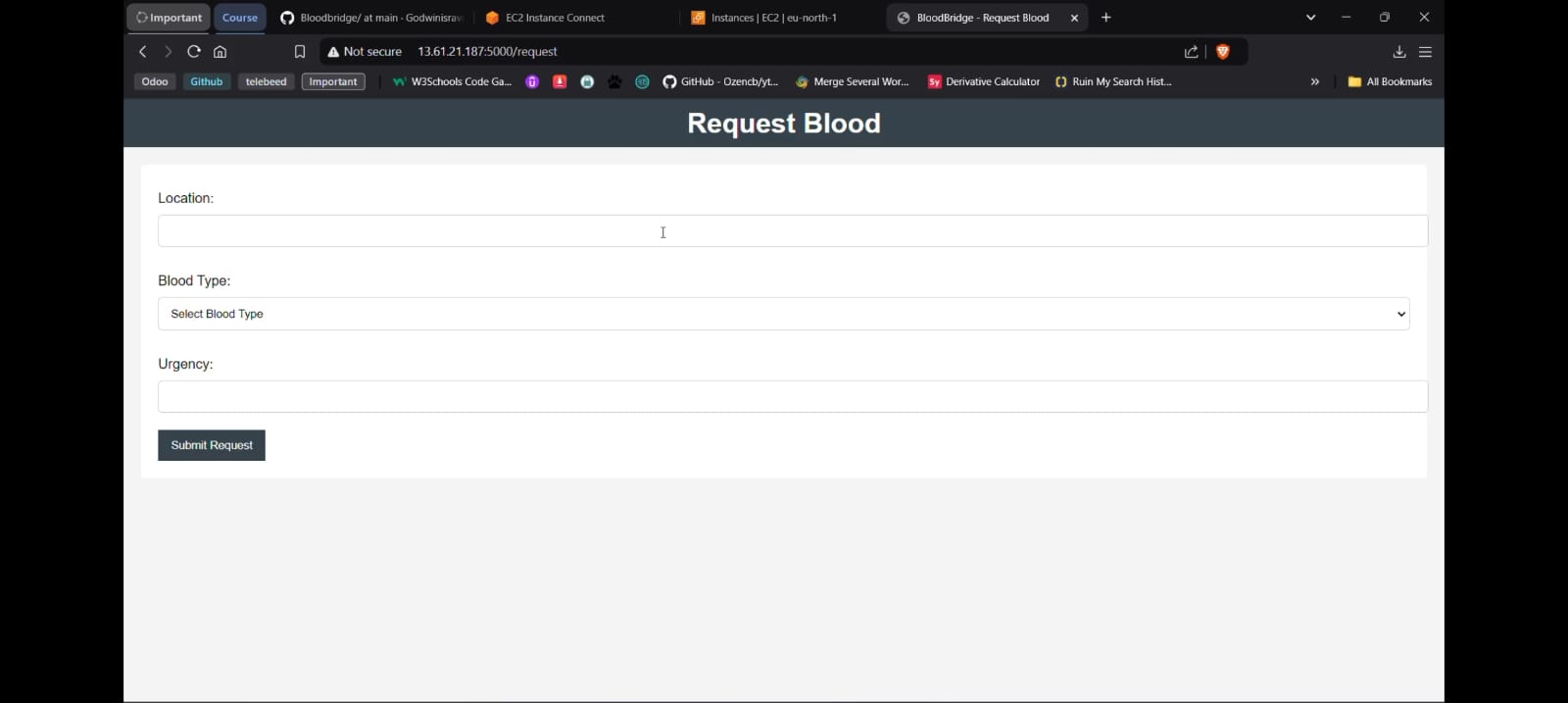
**6.4 Security Testing**  
Conducted HTTPS and data encryption tests to secure sensitive information and applied access control policies, ensuring restricted access to the RDS database only from authorized sources.

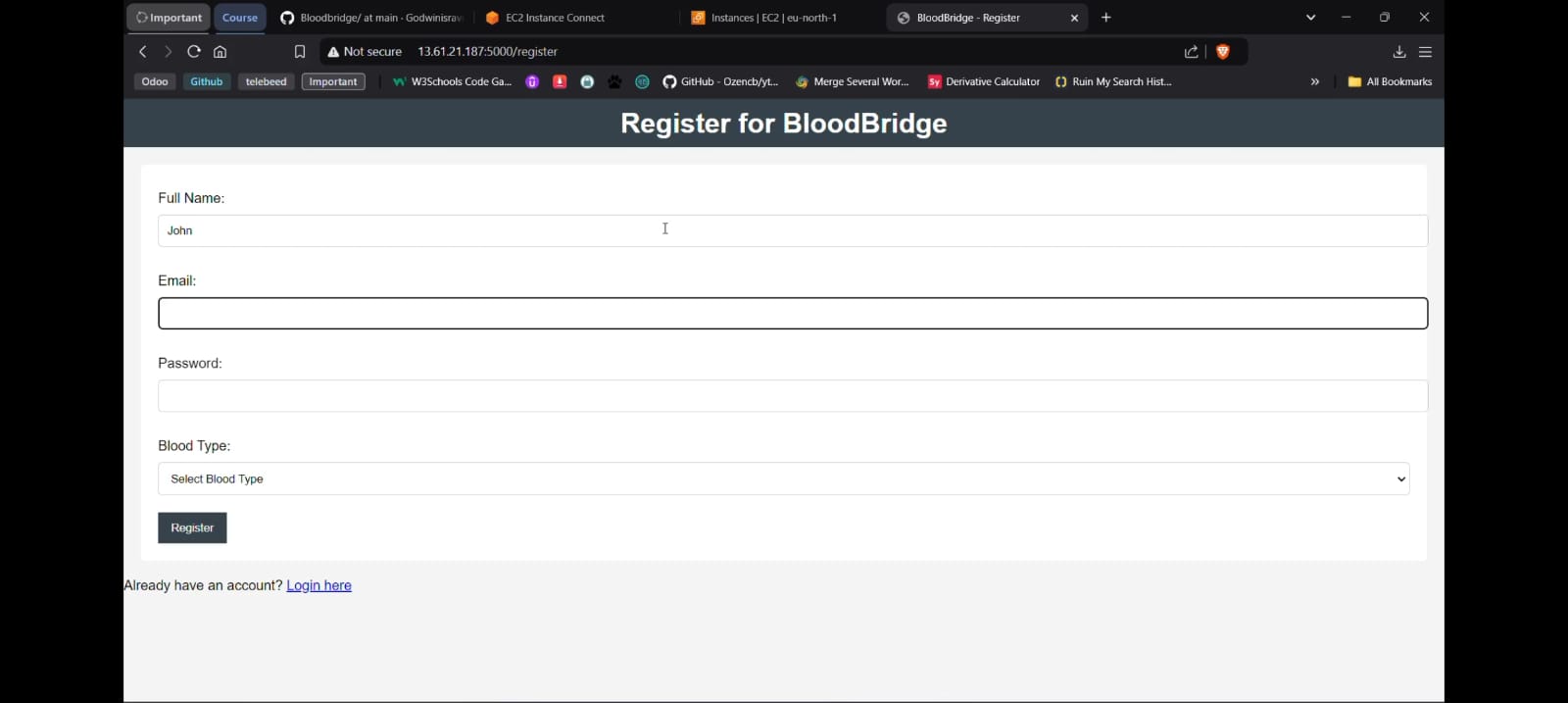
| **Test Type** | **Outcome Summary** |
| --- | --- |
| Functional Tests | Passed for all primary features |
| Load Testing | Stable performance under 500 concurrent users |
| Security Tests | Verified secure HTTPS connections and restricted access |

**SAMPLE USER REGISTRATION SCREEN**

**

**

**

**

**Chapter 7: Conclusion**

**7.1 Summary**  
BloodBridge successfully demonstrates a scalable, cloud-based solution to streamline blood donation logistics. By leveraging AWS infrastructure, the application optimally connects hospitals, donors, and blood banks, enabling prompt response in emergencies and efficient resource management.

**7.2 Future Work**  
Potential future developments include:

* **Integration with mobile notifications**: Expanding to SMS and push notifications.
* **AI Integration**: Using AI algorithms to predict demand based on historical data, assisting hospitals and blood banks in inventory planning.
* **Regional Expansion**: Adapting BloodBridge for deployment in different geographical locations to facilitate a larger network of donors and recipients.
* User Dashboard Interface (placeholder for image) displays how users can navigate requests and view their respective functions based on roles (e.g., donor, hospital admin).

**References**

1. Ariponnammal, S., & Natarajan, S. (1994). Transport Phenomena of Sm Sel – X Asx. *Pramana – Journal of Physics*, 42(1), 421–425.
2. Barnard, R.W., & Kellogg, C. (1980). Applications of Convolution Operators to Problems in Univalent Function Theory. *Michigan Math J.*, 27, 81–94.
3. Shin, K.G., & McKay, N.D. (1984). Open Loop Minimum Time Control of Mechanical Manipulations and its Applications. *Proc. Amer. Contr. Conf.*, San Diego, CA, 1231–1236.