System Requirements Specification

FOR

CanSat Educational Tool

Version 0.1

Sam Moore
Abemelek
Badebo
Bhavya
Patel
Vedant Vyas

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to provide specification for the development of a web-based system for a CanSat Educational Tool. The CanSat Educational Tool is an online platform designed to provide students with hands-on learning experience in satellite and space engineering. By offering resources such as training materials, real-time communication tools, and a virtual Arduino IDE, the platform aims to improve students' technical skills and familiarize them with engineering practices in a simulated CanSat environment.

1.2. Scope

Our software will support three different types of users: Admins, Instructors, and Students. Each user group will have different functionalities ranging from account management and training material creation to collaborative coding and project development. The platform integrates real-time chat and an interactive coding environment to ensure seamless hands-on learning.

1.3. Client information

Client Name: Chanraksmey Lay

Role: Project Creator / Client

Institution: Bucknell University

Field of Study: Mechanical Engineering

Contact information: cl038@bucknell.edu

1.4. System Overview

The system emphasizes user specific dashboards, interactive learning tools, and collaborative features. The virtual Arduino IDE allows students to apply programming concepts to real-world projects, enabling comprehensive skills development in satellite technology.

2. User Requirements

2.1. Summary

The software provides a role-based platform catering to Admins, Instructors, and Students, with functionalities designed to facilitate seamless interactions and hands-on learning. Admins oversee system management, Instructors guide students through projects and provide feedback, and Students engage in collaborative and practical exercises using tools like the Virtual Arduino IDE.

2.2. Requirement Elicitation

To define user requirements, the following elicitation methods were employed:

- **Observation**: Analysis of how project-based learning is conducted in a virtual classroom setting to help replicate similar workflows.
- **Competitor Analysis**: Reviewing features of similar platforms to understand best practices and limitations.
- **Surveys and Questionnaires**: Distributed to potential users to gather insights on desired features, usability, and challenges.

Key takeaways:

- Students prefer interactive, hands-on learning over static content.
- Instructors need tools for efficient time management and communication with students.
- Admins require straightforward account and resource management capabilities.

2.3. High-Level Functionalities

1. **User Role Management:**
   The platform must provide role-specific dashboards and controls based on different user types (Admin, Instructor, Student).
2. **Interactive Learning Tools:**
   The Virtual Arduino IDE must support step-by-step tutorials, real-time code execution, and project-building functionalities.
3. **Communication Features:**
   Real-time chat communication tools should facilitate collaboration and instructor support.
4. **Project Based Learning Support:**
   Resources such as training videos, assignments, and group collaboration features must align with a project-based learning approach.

5. **Time and Activity Tracking:**

   Tools for tracking instructor work hours and student activity logs should be integrated for transparency and accountability.

3. System Stakeholders

3.1 Students

- **Who they are:** The primary users of the system, typically middle and high school age learners who are interested in STEM (Science, Technology, Engineering, and Mathematics) and Aerospace Engineering.

3.2 Parents and Guardians

- **Who they are:** Parents or legal guardians of the students, especially in cases where younger students participate in the CanSat program.

3.3 Educational Institutions

- **Who they are:** Schools**,** typically middle and high schools, and other academic institutions that incorporate the system into their curriculum.

3.4 STEM Educators

- **Who they are:** Teachers and educators specializing in STEM subjects.

3.5 Facilitators/Trainers

- **Who they are:** Individuals responsible for guiding students through workshops, activities, or training programs.

3.6 Avakas Management and Team

- **Who they are:** Internal project management, development team, and stakeholders within the company.

3.7 Suppliers/Partners

- **Who they are:** External suppliers, sponsors, or partner organization that contribute tools, hardware (e.g., Arduino kits), supplies, or funding for the project.

4. System Requirements

4.1. High-Level Functionalities

4.1.1. User Authentication

- **Description**: The system must provide a secure login functionality for all users with role-based access to specific dashboards.
- **Importance**: High
- **Dependencies**: None

4.1.2. Virtual Arduino IDE

- **Description**: The system must provide a web-based Arduino IDE allowing students to engage in interactive coding exercises and project building.
- **Importance**: High
- **Dependencies**: None

4.1.3. Real-Time Communication

- **Description**: The system must facilitate real-time messaging for collaboration and guidance.
- **Importance**: High
- **Dependencies**: None

4.1.4. Resource Management

- **Description**: The system must allow Admins to upload, access, and delete training materials.
- **Importance**: High
- **Dependencies**: 4.1.1

4.1.5. Progress Tracking

- **Description**: The system must be able to track all Students progress and be able to be viewed by Instructors.
- **Importance**: High
- **Dependencies**: None

4.2. Functional Requirements

4.2.1 Admin Functionalities

4.2.1.1 Account Management

- **Description**: Admins must be able to create, update, delete, and assign roles to user accounts.
- **Importance**: High
- **Dependencies**: 4.1.1

4.2.1.2 Activity Monitoring

- **Description**: Admins can view logs of system activities, including user logins, communication, and resource usage.
- **Importance**: Medium
- **Dependencies**: 4.1.1, 4.1.3, 4.2.1.1

4.2.2 Instructor Functionalities

4.2.2.1 Access Materials

- **Description**: Instructors must be able to view training resources uploaded to prepare for teaching.
- **Importance**: High
- **Dependencies**: 4.1.4

4.2.2.2 Time Tracking

- **Description**: Instructors can clock in and out to log work hours, with timestamps visible to Admins for Review.
- **Importance**: Medium
- **Dependencies**: 4.1.1

4.2.2.3 Student Communication

- **Description**: Instructors must be able to send and receive real-time messages with students for guidance.
- **Importance**: High
- **Dependencies**: 4.1.3, 4.2.3.4

4.2.2.4 View Student Progress

- **Description**: Instructors must be able view the progress of the group of students they are assigned to.
- **Importance**: High
- **Dependencies**: 4.1.3, 4.2.3.4

4.2.3 Student Functionalities

4.2.3.1 Access Resources

- **Description**: Students must be able to view training videos, tutorials, and assignments uploaded by Admin.
- **Importance**: High
- **Dependencies**: 4.1.4

4.2.3.2 Virtual Arduino IDE Use

- **Description**: Students must be able to interact with a virtual programming environment to practice programming and complete the assignments
- **Importance**: High
- **Dependencies**: 4.1.2

4.2.3.3 Collaboration Tools

- **Description**: Students can communicate with team members using a real-time chat.
- **Importance**: High
- **Dependencies**: 4.1.3

4.2.3.4 Direct Messaging with Instructor

- **Description**: Students can send messages to instructors to request help or feedback on projects.
- **Importance**: High
- **Dependencies**: 4.1.3, 4.2.2.3

4.3. Non-Functional Requirements

4.3.1 Performance

- **Description**: The system must be able handle up to 200 concurrent users without any loss of performance.
- **Importance**: Medium
- **Dependencies**: None

4.3.2 Security

- **Description**: The system must implement secure authentication protocols (2FA) and encrypt all user data during storage and transmission (HTTPS and AES-256).
- **Importance**: High
- **Dependencies**: 4.1.1

4.3.3 Usability

- **Description**: The system's interface must follow industry-standard UI/UX design principles to ensure ease of navigation, accessibility, and user satisfaction. Specific objectives include:
    - *Navigation:* Users must be able to complete core workflows (e.g., login, IDE usage, resource access) within 3 clicks.
    - *Accessibility:* The system must be compatible with screen readers and meet WCAG 2.1 AA accessibility guidelines.
    - *Error Feedback:* Users must receive clear and actionable feedback for errors (e.g., failed login attempts, file upload errors) within 2 seconds.
- **Importance**: Medium
- **Dependencies**: None

4.3.4 Scalability

- **Description**: The system must be able to support the addition of new features without significant architecture modifications.
- **Importance**: Medium
- **Dependencies**: None

4.3.5 Compatibility

- **Description**: The system must be compatible with modern web browsers (e.g., Chrome, Firefox, Safari) and responsive for mobile access.
- **Importance**: High
- **Dependencies**: None

5. System Models

5.1 Use Case Diagram

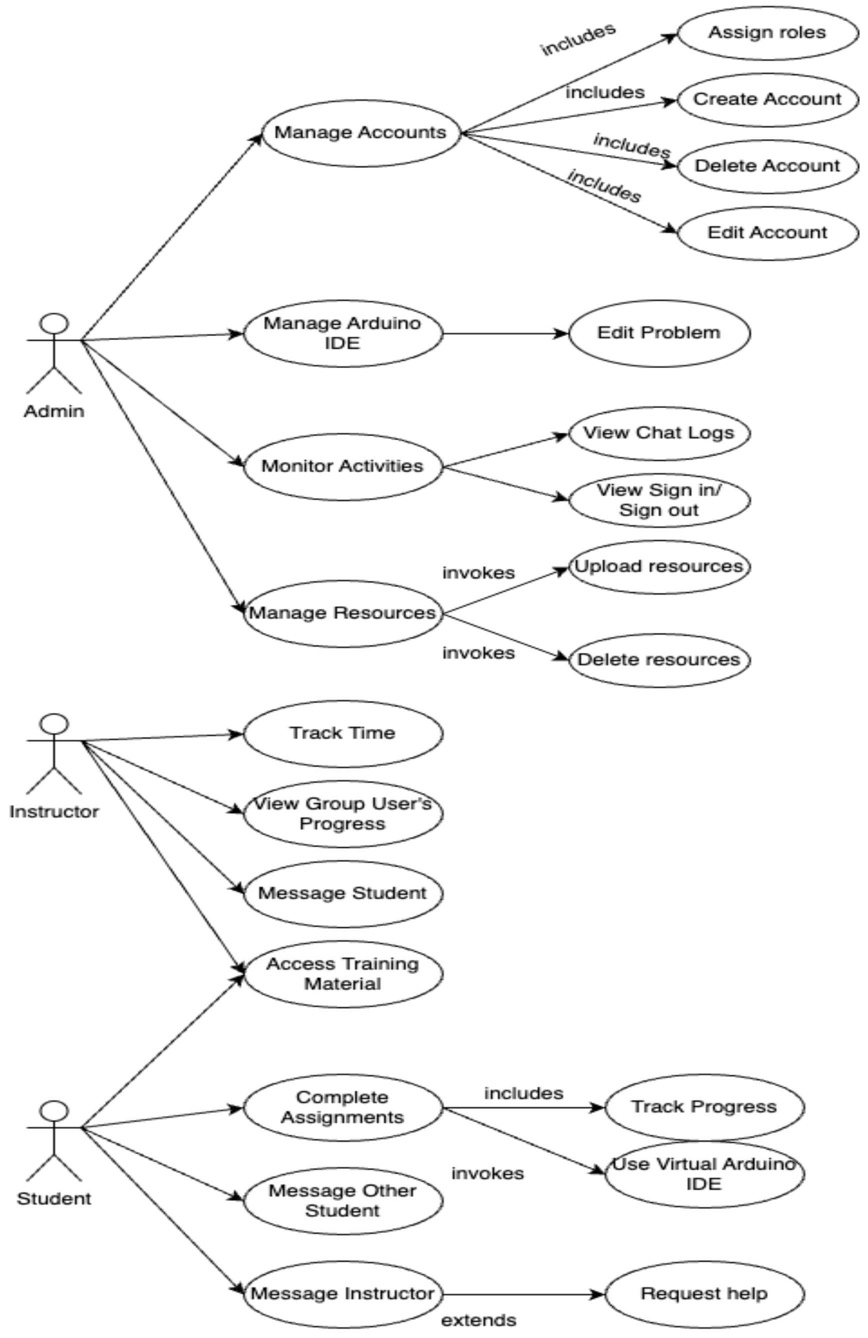


Figure 1: Use Case Diagram

5.2 Sequence Diagrams

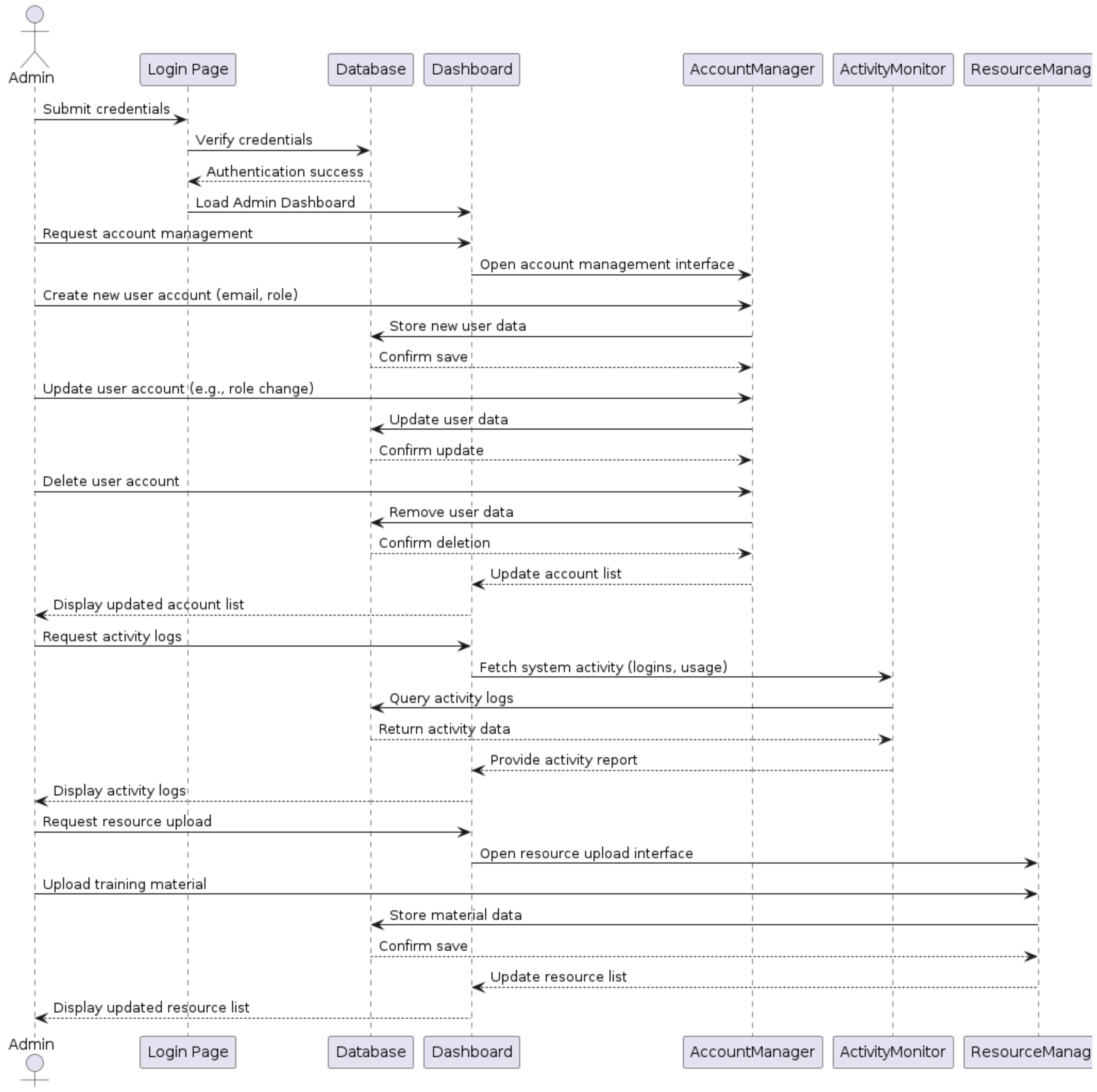


Figure 2: Admin Sequence Diagram

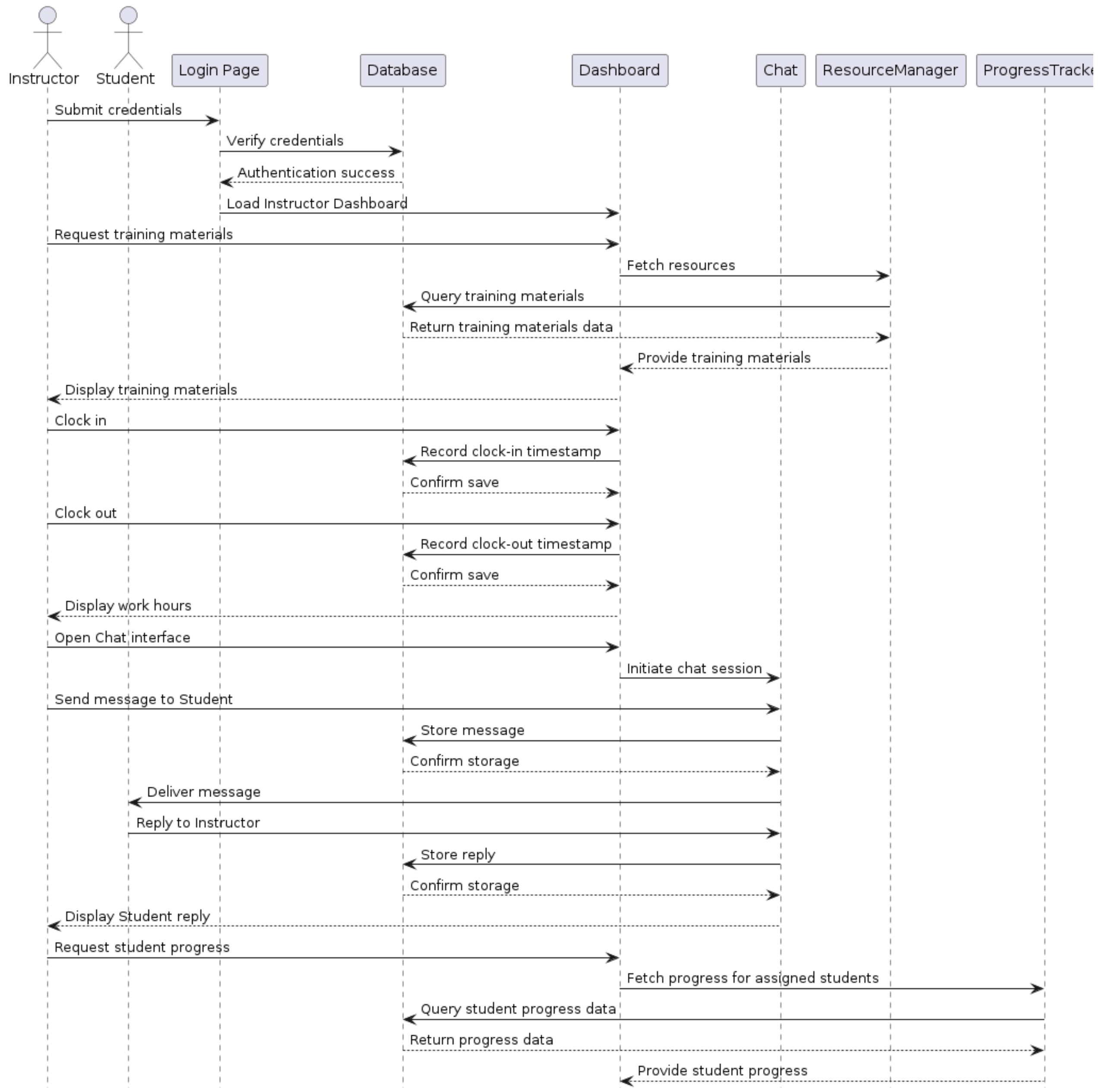


Figure 3: Instructor Sequence Diagram

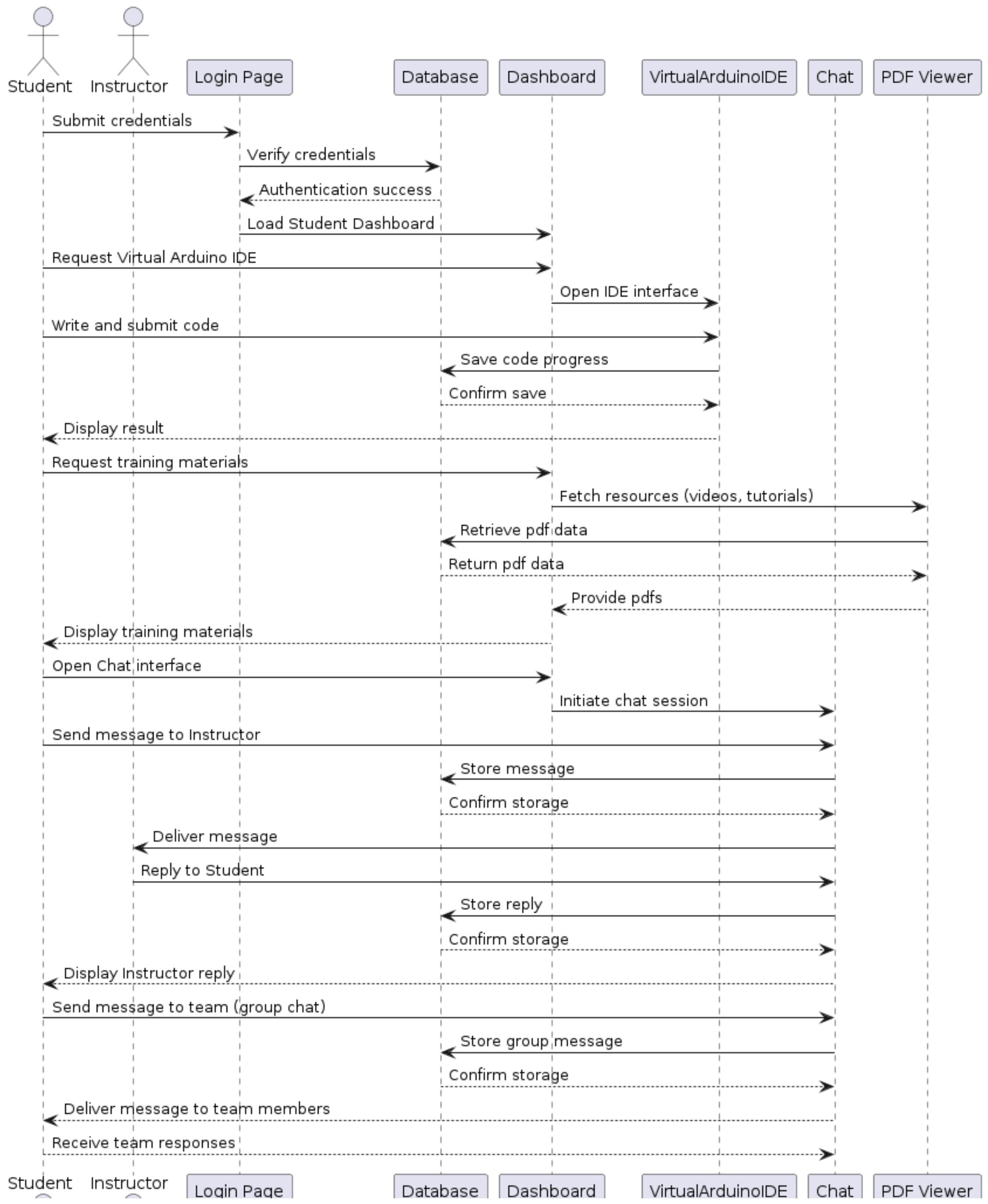
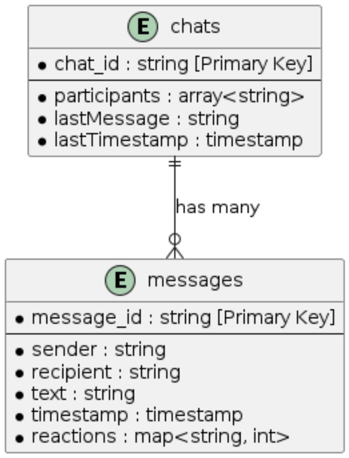


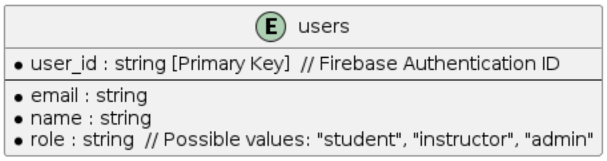
Figure 4: Student Sequence Diagram

5.3 Database Diagrams



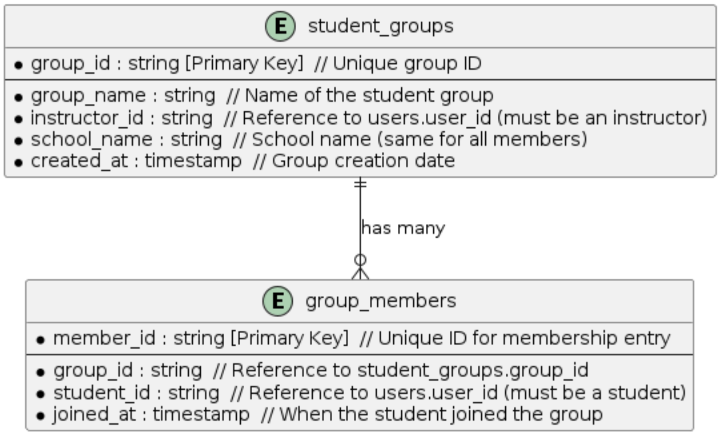
E chats
• chat_id : string [Primary Key]
• participants : array<string>
• lastMessage : string
• lastTimestamp : timestamp
has many
E messages
• message_id : string [Primary Key]
• sender : string
• recipient : string
• text : string
• timestamp : timestamp
• reactions : map<string, int>

Figure 5: Chat collection database schema



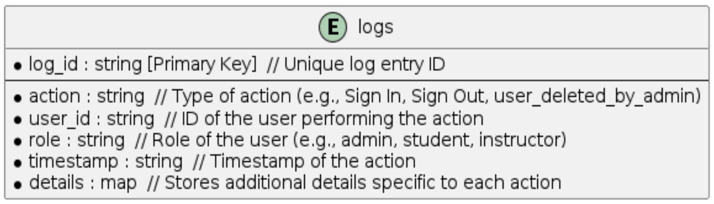
E users
• user_id : string [Primary Key] // Firebase Authentication ID
• email : string
• name : string
• role : string // Possible values: "student", "instructor", "admin"

Figure 6: Users collection database schema



E student_groups
• group_id : string [Primary Key] // Unique group ID
• group_name : string // Name of the student group
• instructor_id : string // Reference to users.user_id (must be an instructor)
• school_name : string // School name (same for all members)
• created_at : timestamp // Group creation date
has many
E group_members
• member_id : string [Primary Key] // Unique ID for membership entry
• group_id : string // Reference to student_groups.group_id
• student_id : string // Reference to users.user_id (must be a student)
• joined_at : timestamp // When the student joined the group

Figure 7: Groups collection database schema



E logs
• log_id : string [Primary Key] // Unique log entry ID
• action : string // Type of action (e.g., Sign In, Sign Out, user_deleted_by_admin)
• user_id : string // ID of the user performing the action
• role : string // Role of the user (e.g., admin, student, instructor)
• timestamp : string // Timestamp of the action
• details : map // Stores additional details specific to each action

Figure 8: Logs collection database schema

5.4 State Diagram

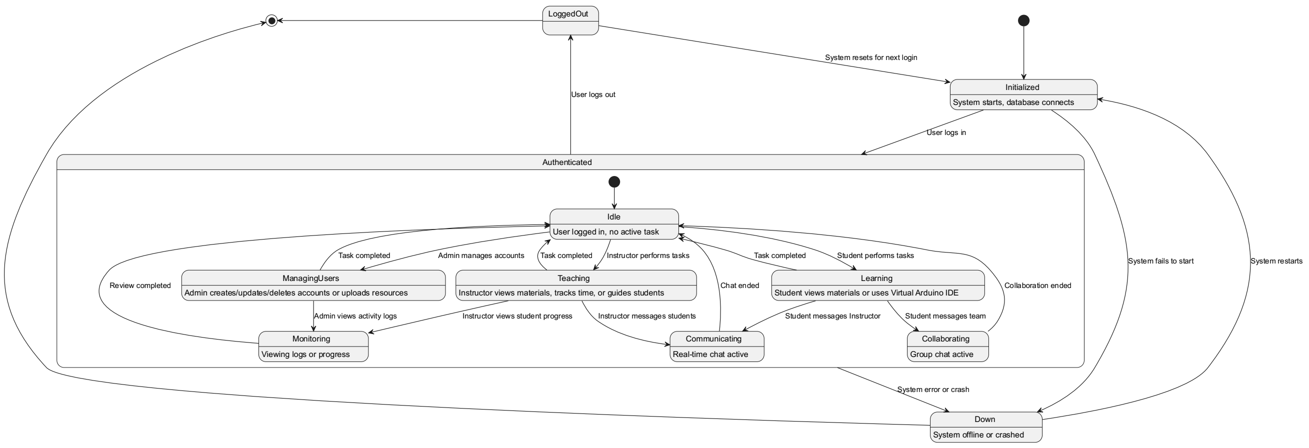

LoggedOut
System resets for next login
Initialized
System starts, database connects
User logs out
User logs in
Authenticated
Idle
User logged in, no active task
Task completed
Admin manages accounts
Task completed
Instructor performs tasks
Task completed
Student performs tasks
ManagingUsers
Admin creates/updates/deletes accounts or uploads resources
Teaching
Instructor views materials, tracks time, or guides students
Learning
Student views materials or uses Virtual Arduino IDE
Review completed
Chat ended
Collaboration ended
System fails to start
System restarts
Admin views activity logs
Instructor views student progress
Instructor messages students
Student messages Instructor
Student messages team
Monitoring
Viewing logs or progress
Communicating
Real-time chat active
Collaborating
Group chat active
System error or crash
Down
System offline or crashed

Figure 9: State Diagram

5.5 Component Diagram

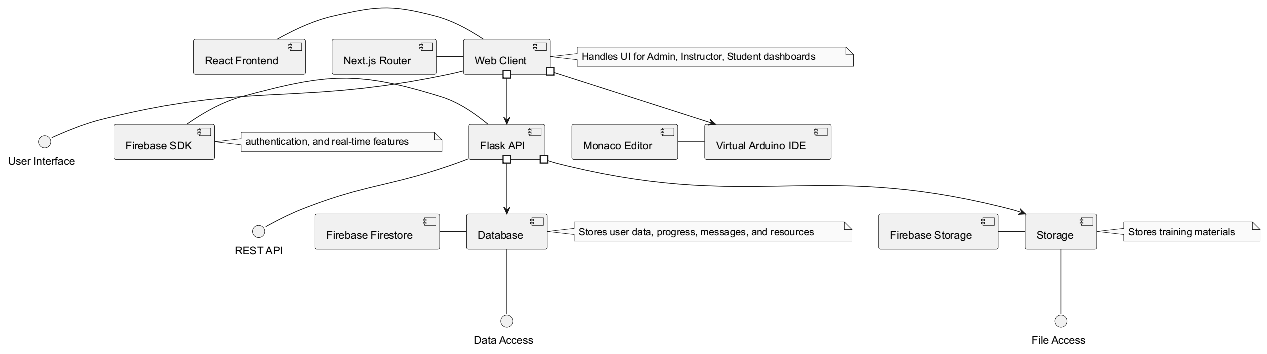


Figure 10: Component Diagram

5.6 Activity Diagrams

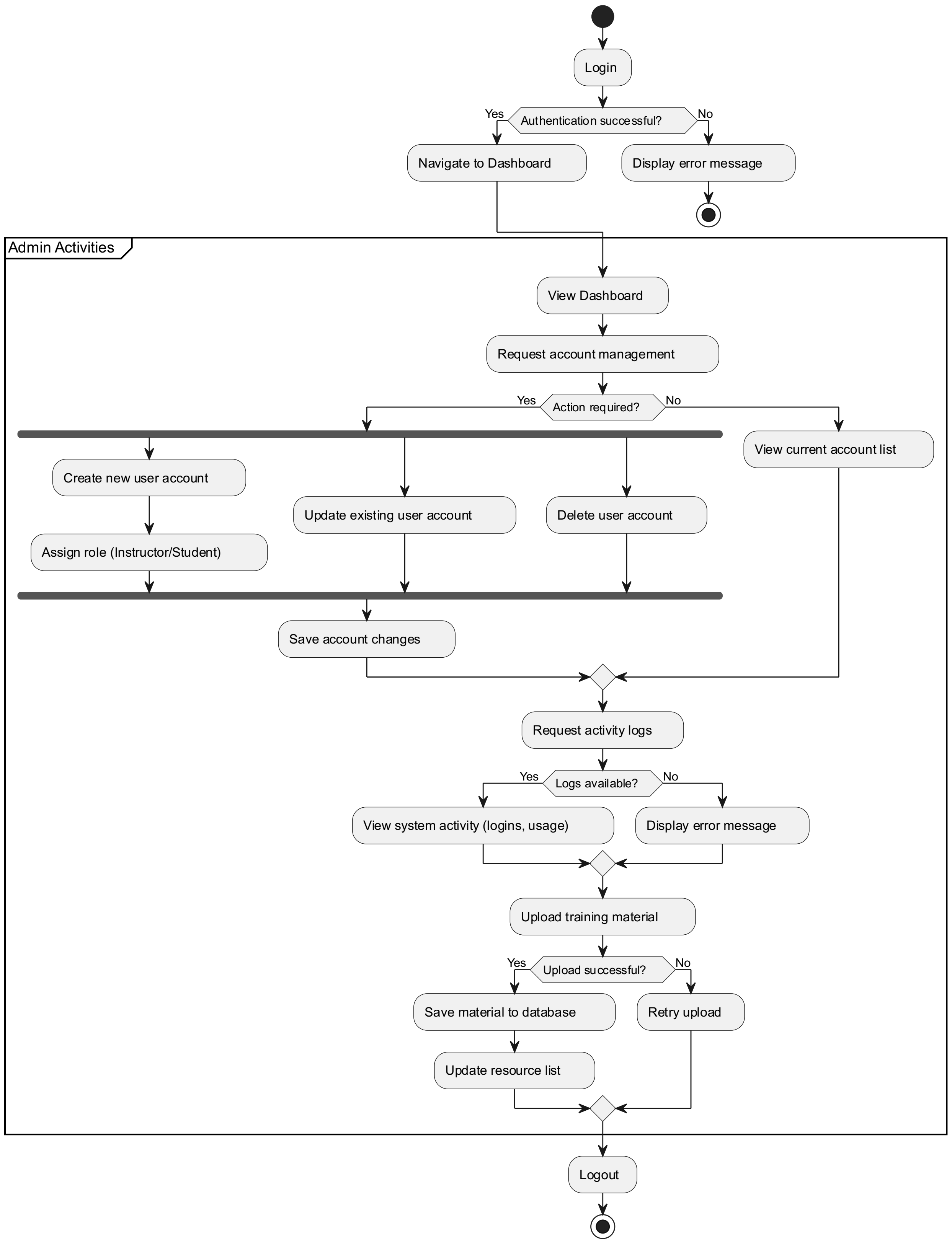


Figure 11: Admin activity diagram

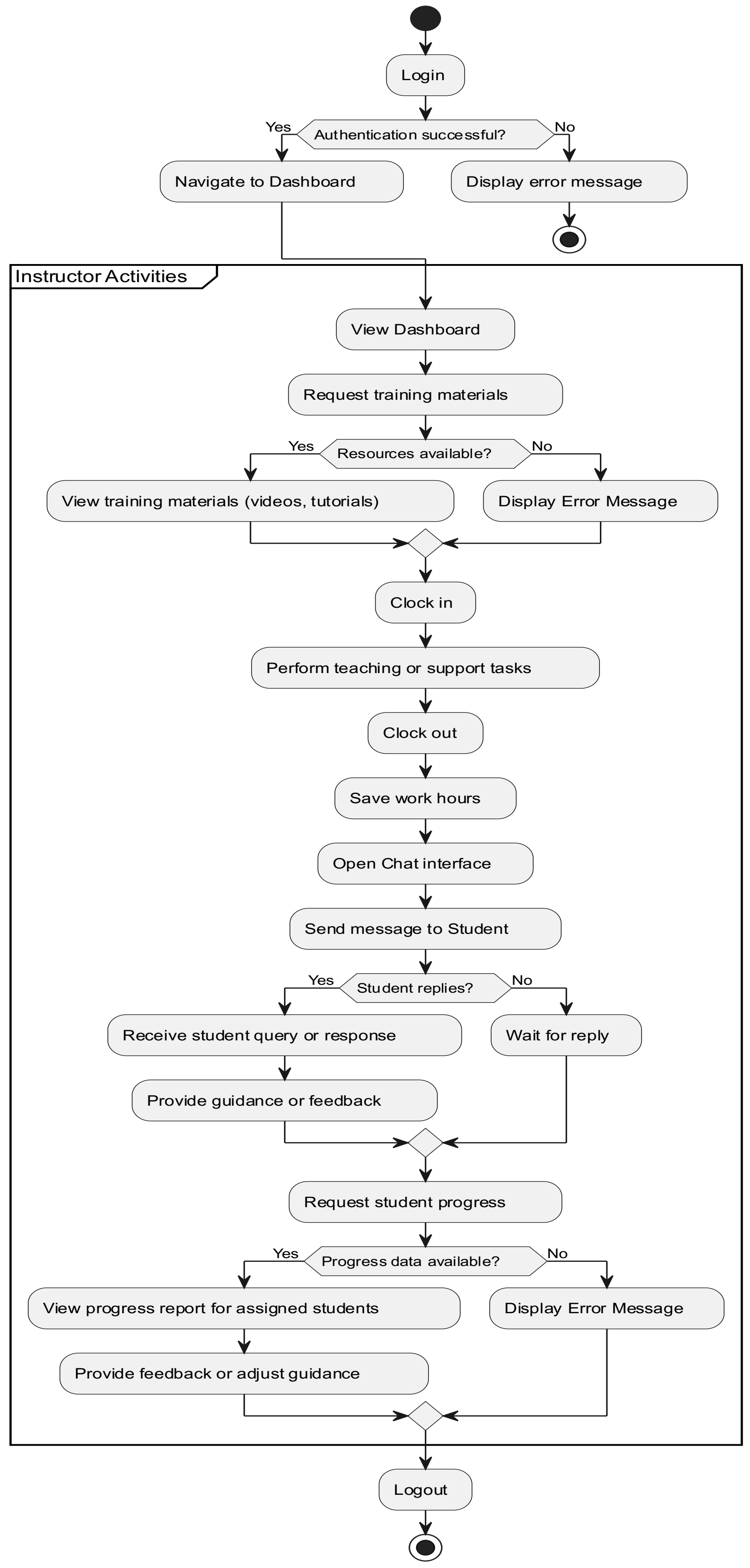


Figure 12: Instructor activity diagram

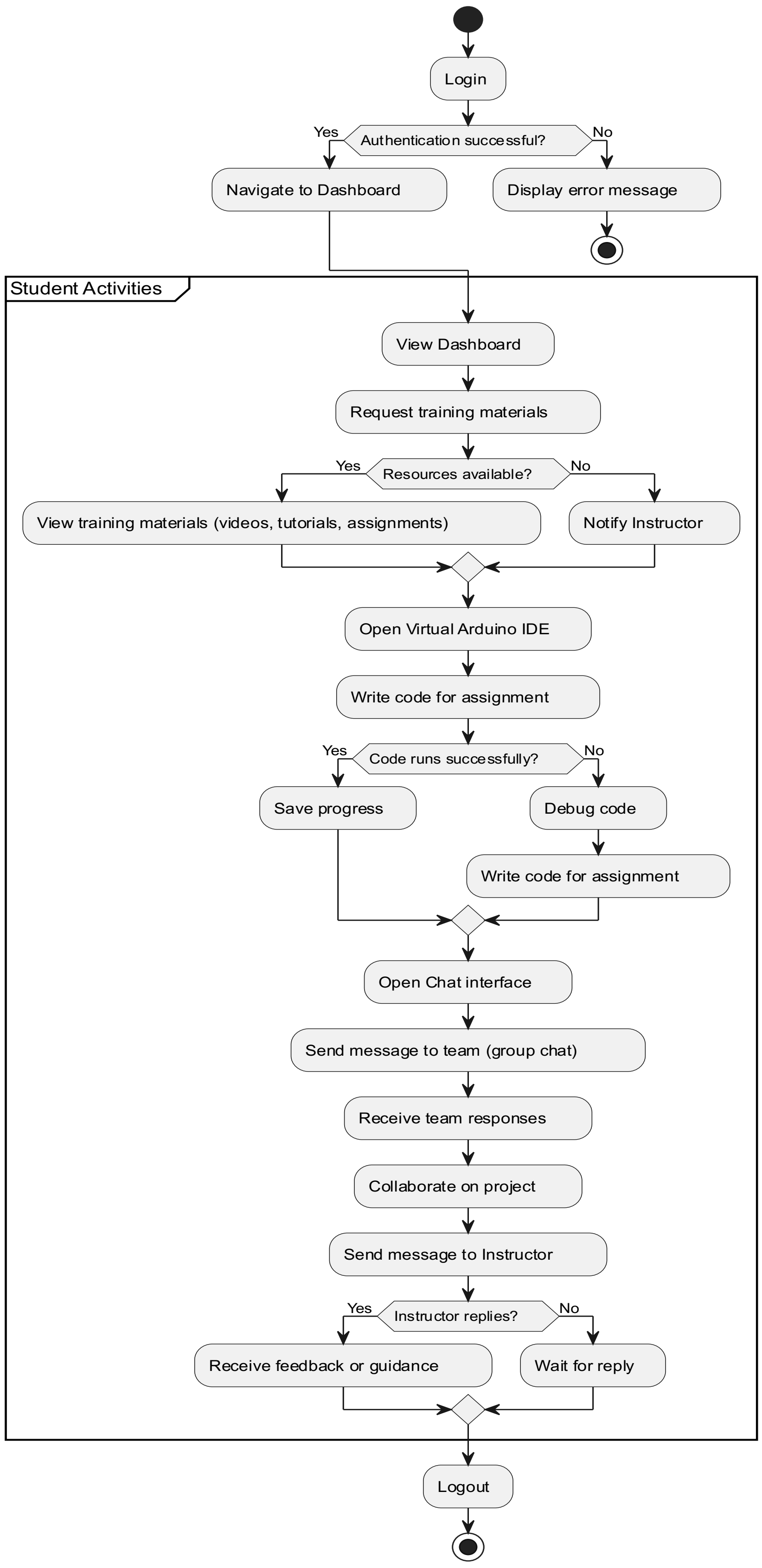


Figure 13: Student activity diagram

6. Software Engineering Tools

6.1. Frontend Development

- **React.js (v: 19) and Next.js (v: 15.1.0)** will be used to design the user interface, creating dynamic and responsive dashboards for Admins, Instructors, and Students. Next.js will handle routing and server-side rending for optimized performance.
- **Tailwindcss (v: 3.0)** will ensure the frontend is responsive, mobile-friendly, and visually cohesive, supporting consistent styling across all pages.

6.2. Backend Development

- **Flask (v3.1.1)** will handle API endpoints, application logic, and integrate third-party services.
- **Firebase SDK (v: 11)** will be used for authentication, directly handling secure login and role-based access control without requiring additional backend logic for managing user sessions. Firebase will also be used for handling real-time features like notifications and chat.
- **Firebase Firestore (v: 4.7.5) Integration** will allow Flask to fetch, update, and manage data stored in Firestore.

6.3. Database

- **Firebase Firestore (v: 4.7.5)** will serve as the primary database for the project. It will handle all structured data such as user roles, account information, training materials, and chat messages.

6.4. Virtual Arduino IDE

- **Monaco Editor (v: 4.7.0)** will be integrated into the frontend as the text editor for the Virtual Arduino IDE, supporting syntax highlighting and real-time coding.
- **Vanilla C++** will simulate Arduino functionality, allowing students to run their code.

6.5. Version Control and Collaboration

- **Git** will track all code changes and manage version control during the development process.

- **Github** will host the project repository, enabling collaboration, code reviews, and issue tracking.

6.6. Testing and QA

- **Selenium (v: 4.9.0)** will automate testing for the frontend user interface to ensure compatibility.
- **Pytest (v: 8.3.4)** will validate backend functionality through unit and integration testing.
- **Jest (29.7.0)** will test React components for consistent behavior.
- **Postman (v: 11.1.14)** will test API endpoints to ensure proper integration between frontend and backend.

6.7. Deployment and Hosting

- **Docker (v: 27)** will containerize the application, making deployment consistent across different environments.
- **Firebase Storage (v: 0.13.7)** will be used to host the training materials.
- **AWS (v: 2.0)** will host the backend.
- **Vercel (v: 39.2.0)** will deploy Next.js frontend with automatic optimizations.

6.8. Project Management

- **Discord (v: 1.0.9184)** will facilitate team communication and integrate updates from tools such as GitHub.

7. Conclusion

The CanSat education tool aims to provide students experience and knowledge about satellite and aerospace engineering by offering a comprehensive, interactive platform. Combining a Next.js and React frontend with a Flask backend and Firebase for database and authentication, the system ensures a scalable, modern architecture. Key features such as the Virtual Arduino IDE, real-time communication tools, and role-based dashboards allow students, instructors, and administrators to engage effectively in a hands-on project-based learning environment.

The project prioritizes functionality, usability, and scalability, with a phased implementation strategy that focuses on developing core features and continuous improvement through client feedback. With its innovative approach to project-based learning, the CanSat education tool has the potential to inspire and prepare future aerospace engineers.

Signature Acknowledgment
The signature line below serves as a formal acknowledgment of the client's review and approval of the requirements outlined in this document. By signing, the client confirms that the system's specifications, functionalities, and constraints have been accurately captured. This approval signifies alignment between the client and development team, allowing the project to progress to the next phase of development.

If any revisions or clarifications are required, they should be addressed prior to signing to ensure all parties have a shared understanding of the project's scope.

Signature: ____ Date: __________________