# **Assignment 1**

## **Members**

* i221110 – Hassan Murtaza
* I220888 – Muhammad Ateeb
* I220933 – Saif Ur Rehman

## **Project Title**

### **Blockchain-Powered Certificate Verification System**

| **The problem of** | fake academic certificates and difficulties in credential verification |
| --- | --- |
| affects | employers, universities, and institution |
| the impact of which | inefficiencies, trust issues, and vulnerability to fraud due to slow, costly, and centralized verification methods |
| a successful solution would be | a blockchain-powered certificate verification system that stores certificate details in a centralized database while keeping the cryptographic hash of the certificate on a blockchain. This ensures tamper-proof verification, prevents fraud, and provides a cost-effective and scalable solution. The system allows institutions and employers to validate degrees quickly and securely, reducing reliance on manual authentication and third-party services. |

### **Feature List**

#### **1. Student Features**

* **Secure Certificate Storage**: Store certificates on a blockchain-backed system for tamper-proof security.
* **QR Code on Certificate**: Generate and embed QR codes linking to certificate verification.
* **Instant Verification**: Enable students to verify their certificates in real-time.
* **Multi-Device Access**: Allow students to retrieve certificates from any device.
* **Revocation & Update Notifications**: Notify students if their certificate status changes.
* **Verification Tracking**: Maintain logs of when and who verified their certificates.

#### **2. University Features**

* **Digital Certificate Issuance**: Provide a platform for universities to issue digital certificates.
* **Cryptographic Hash Generation**: Use SHA-256 to hash certificates for authenticity.
* **Certificate Revocation**: Allow universities to revoke or update incorrect certificates.
* **Admin Dashboard**: Provide a dashboard for managing issued certificates.
* **Student Database Integration**: Connect the certificate issuance system with university databases.
* **Duplicate Certificate Prevention**: Verify existing records before issuing new ones.

#### **3. Employer Features**

* **QR Code Scanning for Verification**: Enable employers to scan QR codes for instant verification.
* **Blockchain Hash Matching**: Validate certificates by checking their hash against blockchain records.
* **User-Friendly Verification Interface**: Provide a simple and intuitive UI for non-technical users.
* **Certificate Detail Display**: Show certificate information upon verification.
* **Verification Timestamping**: Record timestamps of verification requests.

#### **4. System Administrator Features**

* **Certificate Issuance Tracking**: Maintain records of issued, revoked, and verified certificates.
* **Role-Based Access Control (RBAC)**: Restrict system access based on user roles.
* **System Usage Reports**: Generate reports on adoption and certificate verifications.
* **Blockchain Transaction Logging**: Log all blockchain transactions for audit purposes.
* **Automated Data Backups**: Implement periodic backups to prevent data loss.

#### **5. Developer & Scalability Features**

* **Third-Party API for Integration**: Provide REST APIs for external institutions to verify certificates.
* **Modular Codebase**: Use a scalable architecture for future expansions.
* **Automated Testing**: Ensure reliability with unit and integration tests for core functionalities.

**User Stories**

## *For Students*

### **As a student, I want my degree to be stored securely so that I can verify it anytime.**

#### **Acceptance Criteria:**

**Scenario: Uploading a degree certificate**

* **Given** the student is logged into their account
* **When** they upload a degree certificate
* **Then** the system encrypts and stores the certificate securely
* The student receives a confirmation message

**Scenario: Accessing the stored certificate**

* **Given** the student is logged into their account
* **When** they request to view their degree certificate
* **Then** the system decrypts and displays the certificate
* Access is logged with a timestamp

**Scenario: Unauthorized access attempt**

* **Given** an unauthorized user attempts to access the certificate
* **When** they try to view/download the certificate without credentials
* **Then** the system denies access
* And logs the unauthorized attempt

### **As a student, I want my certificate to have a QR code so that I can share it easily.**

#### **Acceptance Criteria:**

**Scenario: Generating a QR code**

* **Given** a certificate is issued
* **When** the QR code is generated
* **Then** the student can share it easily

**Scenario: Scanning a valid QR code**

* **Given** the QR code is scanned by an employer
* **When** the scan is successful
* **Then** the system redirects to the verification page
* And displays the certificate details

**Scenario: Scanning an invalid QR code**

* **Given** a tampered or fake QR code is scanned
* **When** the scan fails
* **Then** the system displays an error: "Invalid QR code"

### **As a student, I want my certificate verification to be instant so that I don’t have to rely on paperwork.**

#### **Acceptance Criteria:**

**Scenario: Instant verification request**

* **Given** the student shares their certificate
* **When** an employer initiates verification
* **Then** the system validates the certificate in <2 seconds
* And returns a "Valid" status

**Scenario: Verification of revoked certificate**

* **Given** the certificate is revoked
* **When** verification is requested
* **Then** the system returns a "Revoked" status
* And displays the revocation date

### **As a student, I want to access my certificate from any device so that I can retrieve it anytime.**

#### **Acceptance Criteria:**

**Scenario: Access via mobile device**

* **Given** the student uses a mobile browser
* **When** they log into their account
* **Then** the certificate is displayed in a mobile-friendly format

**Scenario: Access via desktop**

* **Given** the student uses a desktop browser
* **When** they log into their account
* **Then** the certificate is displayed in high resolution

### **As a student, I want to receive notifications if my certificate is updated or revoked so that I stay informed.**

#### **Acceptance Criteria:**

**Scenario: Notification for revocation**

* **Given** the university revokes a certificate
* **When** the revocation is processed
* **Then** the student receives an email/SMS notification
* And the notification includes the revocation reason

**Scenario: Notification for updates**

* **Given** the certificate is updated (e.g., corrected typos)
* **When** the update is saved
* **Then** the student receives an email/SMS with a version history link

### **As a student, I want to track my certificate verification requests so that I know when employers verify my credentials.**

#### **Acceptance Criteria:**

**Scenario: Tracking verification requests**

* **Given** an employer verifies the student’s certificate
* **When** the verification is complete
* **Then** the student’s dashboard shows the employer’s name and timestamp

**Scenario: No verification history**

* **Given** no employers have verified the certificate
* **When** the student checks their dashboard
* **Then** the system displays: "No verification requests yet"

## *For Universities*

### **As a university, I want to issue digital certificates so that students can store them securely.**

#### **Acceptance Criteria:**

**Scenario: Issuing a certificate**

* **Given** the student’s data is validated
* **When** the university approves the certificate
* **Then** a digital certificate is generated
* And the student receives it via email

**Scenario: Duplicate certificate prevention**

* **Given** a certificate already exists for the student
* **When** the university tries to reissue it
* **Then** the system blocks duplication
* And alerts: "Certificate already issued"

### **As a university, I want to generate a cryptographic hash for each certificate so that I can ensure its authenticity.**

#### **Acceptance Criteria:**

**Scenario: Generating a hash**

* **Given** a certificate is issued
* **When** the certificate is saved
* **Then** a SHA-256 hash is generated and stored on the blockchain

**Scenario: Tampered certificate detection**

* **Given** the certificate’s hash is modified
* **When** verification is requested
* **Then** the system flags a mismatch
* And returns "Tampered Certificate"

### **As a university, I want to revoke incorrect certificates so that only valid credentials remain.**

#### **Acceptance Criteria:**

**Scenario: Revoking a certificate**

* **Given** the university identifies an incorrect certificate
* **When** revocation is confirmed
* **Then** the certificate’s status is updated to "Revoked"
* And the blockchain record is marked invalid

**Scenario: Revoking an already revoked certificate**

* **Given** the certificate is already revoked
* **When** the university tries to revoke it again
* **Then** the system alerts: "Certificate already revoked"

### **As a university, I want an admin dashboard to manage issued certificates so that I can efficiently track them.**

#### **Acceptance Criteria:**

**Scenario: Filtering certificates by status**

* **Given** the admin selects "Revoked" in the dashboard
* **When** they apply the filter
* **Then** only revoked certificates are displayed

**Scenario: Bulk actions**

* **Given** the admin selects multiple certificates
* **When** they choose "Export to CSV"
* **Then** a CSV file with selected certificates is downloaded

### **As a university, I want to integrate certificate issuance with my existing student database so that the process is seamless.**

#### **Acceptance Criteria:**

**Scenario: Automatic data sync**

* **Given** a student’s data is updated in the university database
* **When** the certificate is generated
* **Then** the system pulls the latest data automatically

**Scenario: Data mismatch error**

* **Given** the student’s data in the database is incomplete
* **When** the university tries to issue a certificate
* **Then** the system blocks issuance
* **And** alerts: "Incomplete student data"

### **As a university, I want to verify certificates instantly before issuing duplicates so that I prevent fraudulent claims.**

#### **Acceptance Criteria:**

**Scenario: Pre-issuance verification**

* **Given** a student requests a duplicate certificate
* **When** the university initiates verification
* **Then** the system checks the blockchain for existing hashes
* **And** blocks duplicates if matches are found

#### *For Employers*

### **As an employer, I want to scan a QR code to verify a candidate’s certificate so that I can confirm its legitimacy.**

#### **Acceptance Criteria:**

##### **Scenario: Valid QR code scan**

* **Given** the QR code is scanned
* **When** the certificate is valid
* **Then** the system displays "Verified"
* **And** shows the candidate’s name and issuance date

##### **Scenario: Expired QR code**

* **Given** the QR code is scanned
* **When** the certificate is expired
* **Then** the system displays "Expired: Contact Institution"

### **As an employer, I want the system to instantly check if the certificate hash matches the blockchain record so that I can detect fraud.**

#### **Acceptance Criteria:**

##### **Scenario: Hash validation success**

* **Given** the certificate hash is checked
* **When** it matches the blockchain record
* **Then** the system displays "Authentic Certificate"

##### **Scenario: Hash validation failure**

* **Given** the certificate hash is checked
* **When** it does not match the blockchain
* **Then** the system displays "Fraud Alert: Hash Mismatch"

### **As an employer, I want a simple verification interface so that I don’t need technical knowledge to use it.**

#### **Acceptance Criteria:**

##### **Scenario: One-click verification**

* **Given** the employer visits the verification page
* **When** they upload the certificate or scan the QR code
* **Then** results are displayed in plain language (e.g., "Valid/Invalid")

### **As an employer, I want to see certificate details when verifying so that I can confirm relevant credentials.**

#### **Acceptance Criteria:**

##### **Scenario: Viewing certificate details**

* **Given** the certificate is verified
* **When** the employer clicks "View Details"
* **Then** the system displays the student’s name, degree, and issuance date

### **As an employer, I want to receive a verification timestamp so that I know when the certificate was checked.**

#### **Acceptance Criteria:**

##### **Scenario: Timestamp generation**

* **Given** the verification is complete
* **When** the employer requests a report
* **Then** the PDF/email includes the verification date and time

#### *For System Administrators*

### **As an admin, I want to track certificate issuance history so that I can audit system activity.**

#### **Acceptance Criteria:**

##### **Scenario: Exporting audit logs**

* **Given** the admin selects a date range
* **When** they export the logs
* **Then** a CSV file with timestamps, issuers, and student IDs is generated

### **As an admin, I want to restrict access to university personnel so that security is maintained.**

#### **Acceptance Criteria:**

##### **Scenario: Unauthorized access attempt**

* **Given** a non-admin user tries to access the admin dashboard
* **When** they attempt to log in
* **Then** the system blocks access
* **And** logs the IP address

### **As an admin, I want to generate system usage reports so that I can monitor adoption rates.**

#### **Acceptance Criteria:**

##### **Scenario: Monthly usage report**

* **Given** the admin selects "Monthly Report"
* **When** they generate it
* **Then** the PDF includes total verifications, issuances, and active users

### **As an admin, I want to integrate blockchain transactions with a log system so that every action is traceable.**

#### **Acceptance Criteria:**

##### **Scenario: Blockchain transaction logging**

* **Given** a certificate is issued or revoked
* **When** the action is completed
* **Then** the blockchain transaction ID is saved in the system log

### **As an admin, I want a backup system for centralized data so that certificate records are never lost.**

#### **Acceptance Criteria:**

##### **Scenario: Automated daily backup**

* **Given** the system is online
* **When** the clock reaches 12:00 AM
* **Then** all data is backed up to a secure cloud server

#### *For Developers & Future Expansions*

### **As a developer, I want an API for third-party integrations so that other institutions can connect to our verification system.**

#### **Acceptance Criteria:**

##### **Scenario: API authentication**

* **Given** a third-party system sends an API request
* **When** they provide a valid API key
* **Then** the system returns the requested certificate data

### **As a developer, I want a modular codebase so that new features can be added without breaking existing functionality.**

#### **Acceptance Criteria:**

##### **Scenario: Adding a new verification method**

* **Given** the codebase is modular
* **When** a developer adds a "Face Recognition" module
* **Then** existing certificate issuance/verification features remain functional

### **As a developer, I want automated tests for certificate issuance and verification so that system reliability is ensured.**

#### **Acceptance Criteria:**

##### **Scenario: Automated testing for verification**

* **Given** the system has automated tests
* **When** a new verification method is added
* **Then** the tests confirm that certificate issuance and verification remain functional

**Roles and Responsibilities**

1. **Product Owner:** Hassan Murtaza – Defines project vision and backlog.
2. **Scrum Master:** Muhammad Ateeb – Ensures Agile workflow and tracks progress.
3. **Scrum Team:** Saif Ur Rehman – Implements features and documentation.

# **Team Agreement**

## **Communication Methods**

To maintain clear and effective communication, we will use the following methods:

* **Primary Communication:** WhatsApp Messenger (for quick discussions and updates).
* **Email:** For formal communication, sharing reports, and submission-related discussions.
* **Phone Calls:** Only for urgent matters.
* **Trello Board:** Used for tracking tasks, backlog items, and sprint progress.
* **Google Drive:** Used for sharing documents, meeting notes, and project artifacts.

## **Communication Response Times**

* **WhatsApp:** Within **2 hours**.
* **Email:** Within **12 hours**.
* **Phone Calls:** Immediate response if possible; otherwise, call back within **1 hour**.
* **Trello Updates:** Must be checked daily, and responses to task updates should be within **24 hours**.

## **Meeting Attendance & Participation**

* **Regular Meetings:** Twice a week (**Monday & Thursday at 8 PM**).
* **Mandatory Attendance:** All members must attend unless there is a valid reason.
* **Absence Protocol:** Inform the group at least **4 hours in advance**.

### **Running Meetings**

* **Meeting Type:** Online via **Google Meet** unless a face-to-face meeting is necessary.
* **Meeting Host:** **Scrum Master (Muhammad Ateeb)** will lead the discussion.
* **Minutes of Meeting (MoM):** **Saif Ur Rehman** will document key points and decisions.
* **Meeting Duration:** Maximum **1 hour**.
* **Agenda Sharing:** The **Product Owner (Hassan Murtaza)** will share the meeting agenda **6 hours before the meeting**.

### **Meeting Preparation**

* Members must review **Trello tasks** and bring updates before each meeting.
* If required, members should prepare **code snippets, reports, or research** before the meeting.
* If someone is **unprepared**, they must inform the team before the meeting.

## **Version Control**

* **GitHub Repository:** Will be used for code management.
* **What to Commit?** Only **working, reviewed code**; no personal files or experimental code.
* **Commit Messages:** Must be clear and descriptive (e.g., *"Implemented login feature with validation"*).
* **Branching Strategy:**
  + main – Only **stable, production-ready code**.
  + develop – **Active development** branch.
  + **Feature branches** (feature/login, feature/database) – Used for individual tasks.

## **Division of Work**

* **Product Owner (Hassan Murtaza):** Defines requirements, user stories, and backlog.
* **Scrum Master (Muhammad Ateeb):** Assigns tasks, monitors workflow, and facilitates meetings.
* **Scrum Team (Saif Ur Rehman):** Implements features, documents processes, and performs testing.

### **Task Assignments**

* The **Scrum Master** will propose task assignments, and the team will agree collectively.
* Work will be divided based on **expertise, availability, and workload**.
* A **Trello board** will be maintained to track individual progress.

## **Submitting Assignments**

* **Internal Submission Deadline:** **24 hours before the official deadline**.
* **Who Submits?** **Scrum Master (Muhammad Ateeb)** is responsible for final submission.

### **Pre-Submission Review**

* **Saif Ur Rehman** will review **documentation**.
* **Hassan Murtaza** will verify that all features **match the user stories**.
* **Muhammad Ateeb** will conduct a final **quality check**.

## **Contingency Planning**

To handle unexpected issues, we have established the following plans:

### **1. If a team member drops out:**

* Responsibilities will be **redistributed** among the remaining members.
* The **Scrum Master** will immediately **inform the instructor** for further guidance.

### **2. If a team member consistently misses meetings:**

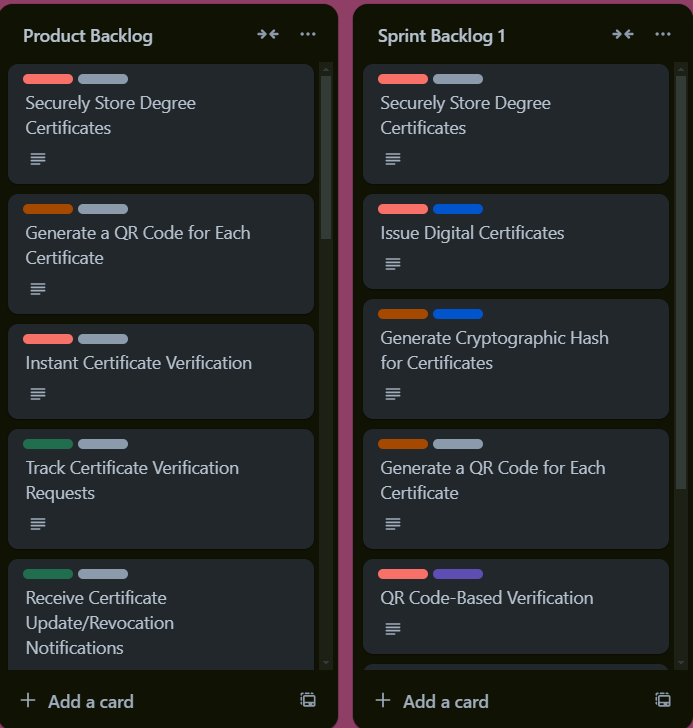
* The **Scrum Master** will personally **follow up**.
* If absenteeism continues, the issue will be **escalated to the Product Owner**.
* If unresolved, a **formal complaint may be raised**.

### **3. If a team member is academically dishonest:**

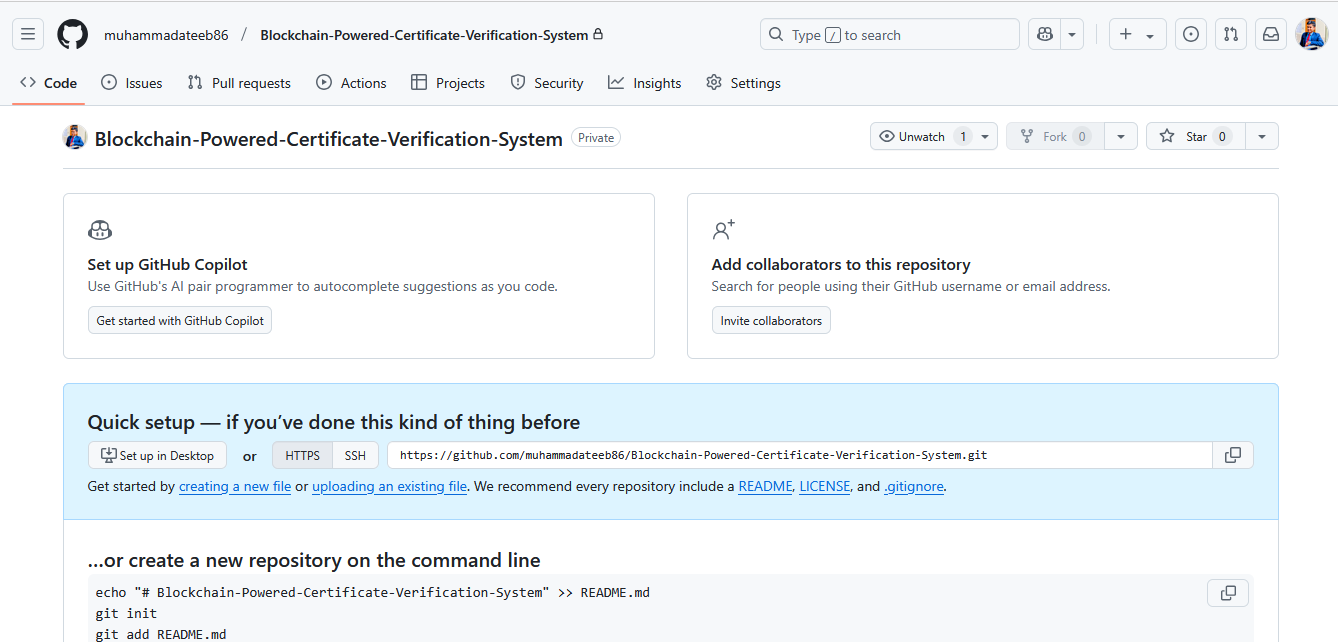
* The team will hold a **meeting to discuss the issue**.
* If the issue persists, it will be **reported to the instructor**.
* The member will **not receive credit** for dishonest work.

## **Final Note**

This agreement ensures **transparency, accountability, and efficiency** in our Agile workflow. Each team member is responsible for following these guidelines to contribute effectively to our project.

**Screenshots  
**

**Github Repository**

****

**Github Link**[**https://github.com/muhammadateeb86/Blockchain-Powered-Certificate-Verification-System**](https://github.com/muhammadateeb86/Blockchain-Powered-Certificate-Verification-System)