**FRAUD DETECTION IN LOCAL CLASSROOM ASSIGNMENT**

**REVIEW - 0**

CB.EN.U4CSE22507

CB.EN.U4CSE22511

CB.EN.U4CSE22538

CB.EN.U4CSE22545

CB.EN.U4CSE22556

**ROLES**

(1) Product Owner : 22556

(2) SCRUM Master : 22538

(3) Developers : 22511 & 22545

(4) Tester : 22507

**SCOPE**

The Fraud Detection System aims to provide automated detection of suspicious activities and potential fraud in local classroom assignments. It incorporates machine learning models to analyze patterns, flag anomalies, and assist educators in ensuring integrity in assignments and evaluations.

**BASIC PROJECT TIMELINE**

|  |  |
| --- | --- |
| Week 0 | Activities |
| Week 1 | Sprint 0: Feasibility study, initial planning, and JIRA setup. |
| Week 2 | Narration of user stories and sprint planning. Begin Sprint 1 development. |
| Week 3 | Continue Sprint 1. Implement 30% of features and draft test cases. |
| Week 4 | Sprint 1 review and retrospective. Start Sprint 2 implementation. |
| Week 5 | Continue Sprint 2: Complete feature implementation and code analysis. |
| Week 6 | Conduct integration testing and finalize system for Sprint 2. |
| Week 7 | Start Sprint 3: Feature optimization, system analysis, and addressing feedback. |
| Week 8 | Continue Sprint 3: Finalize feature set, implement bug fixes, and draft final documentation. |
| Week 9 | Sprint 3 review and retrospective. Start Sprint 4 planning. |
| Week 10 | Begin Sprint 4: Final code enhancements, review features, and prepare for testing. |
| Week 11 | Sprint 4 continued: Conduct system testing and address any issues. |
| Week 12 | Sprint 4 review and retrospective. Prepare deployment environment and documentation. |
| Week 13 | Start Sprint 5: Final deployment preparation, deployment on staging environment. |
| Week 14 | Sprint 5: Conduct user acceptance testing (UAT) and address any user feedback. |
| Week 15 | Final project delivery, wrap-up documentation, and team presentations. |

**PLANNING**

**Basic Project Timeline**:

* + 1. Phase 1: Requirement Gathering (Week 1-3)
    2. Phase 2: Design and Modelling (Week 4-6)
    3. Phase 3 : Development and Implementation (Week 6-10)
    4. Phase 3: Testing (Week 10-12)
    5. Phase 4 : Deployment(Week 12-15)

**EPICS & USER STORIES**

(1) User Authentication and Authorization

[1.1] As a **user**, I want to register with my personal details, so that I can create a secure account to access the system.

Tasks:

(1) Design a registration form to collect user details (e.g., name, email, password).

(2) Implement validation for mandatory fields and password complexity.

(3) Encrypt passwords before storing them in the database.

(4) Send a confirmation email with a verification link.

(5) Test the registration flow for edge cases and errors.

[1.2] As a **user**, I want to log in using my email and password, so that I can securely access my account.

Tasks:

(1) Create a login form with fields for email and password.

(2) Validate user credentials against stored data.

(3) Implement session management to maintain user state.

(4) Add error messages for incorrect credentials.

(5) Test the login process for performance and security.

[1.3] As a **user**, I want to reset my password if I forget it, so that I can regain access to my account.

Tasks:

(1) Provide a "Forgot Password" option on the login screen.

(2) Implement a process to send a password reset link via email.

(3) Create a password reset form with validation.

(4) Update the user's password in the database upon successful reset.

(5) Test the password reset functionality for usability and security.

[1.4] As an **administrator**, I want to assign roles (teacher, student, administrator) to users, so that I can manage user permissions and access control.

Tasks:

(1) Design a role assignment interface within the admin panel.

(2) Implement a role-based access control (RBAC) system.

(3) Restrict access to specific features based on user roles.

(4) Provide feedback messages when access is denied.

(5) Test role assignments for consistency and correctness.

[1.5] As a **user**, I want to log out of my account, so that I can ensure my session is securely closed.

Tasks:

1. Add a "Logout" button to the user interface.
2. Clear session data and authentication tokens upon logout.
3. Redirect the user to the login screen after logging out.
4. Prevent access to restricted pages after logout.
5. Test the logout process for various scenarios and session states.

(2) Fraud Detection System

[2.1] User Story 1: As an **instructor**, I want a system to flag suspicious submissions so that I can review potential fraud.

Tasks:

1. Research and implement plagiarism detection algorithms.

2. Develop a user interface to display flagged submissions.

3. Create a database schema to store flagged records.

4. Integrate the plagiarism detection engine with assignment uploads.

5. Test the system with sample assignments for false positives/negatives.

[2.2] User Story 2: As a **student**, I want clear feedback if my submission is flagged so that I can correct any issues.

Tasks:

1. Design a notification system for flagged submissions.

2. Build a feature to display reasons for flagging (e.g., duplicate content, late submission).

3. Implement a resubmission process with tracking.

4. Add a feature to compare flagged submissions with others.

5. Write and deploy a help section for students to understand flags.

[2.3] User Story 3: As an **administrator**, I want to monitor system performance so that I can ensure smooth operation.

Tasks:

1. Develop dashboards for real-time system performance metrics.

2. Create logs for flagged submissions and detection activities.

3. Implement a feature for administrators to adjust detection thresholds.

4. Automate report generation on flagged activity.

5. Set up alerts for system errors or downtime.

[2.4] User Story 4: As an **instructor**, I want the system to detect collusion between students so that I can maintain academic integrity.

Tasks:

1. Develop a feature to compare assignment content between multiple submissions.

2. Research and apply text-matching algorithms for similarity detection.

3. Build visualizations to highlight suspicious patterns.

4. Test and refine algorithms for accuracy.

5. Create an admin panel for instructors to override detection results.

[2.5] User Story 5: As a **student**, I want assurance that my privacy is respected so that I can trust the system.

Tasks:

1. Implement secure data encryption for stored submissions.

2. Add privacy policies and terms of use for students to accept.

3. Limit access to flagged data to authorized users only.

4. Perform regular audits of the system’s access logs.

5. Build a feature for students to request deletion of old submissions.

(3) Reporting and Notifications

[3.1] User Story 1: As a **teacher**, I want to generate a summary report of flagged assignments so that I can review suspicious activities easily.

Tasks:

1. Create a button on the interface to generate the report.
2. Fetch flagged assignment data from the system.
3. Organize the data into a readable tabular format.
4. Add basic statistics (e.g., total flagged assignments, percentage of flagged assignments).
5. Save the report and provide a download option.

[3.2] User Story 2: As a **teacher**, I want to receive notifications about flagged assignments so that I can take timely action.

Tasks:

1. Set up a system to detect flagged assignments in real-time.
2. Trigger a notification when an assignment is flagged.
3. Display notifications on the teacher’s dashboard.
4. Add a "mark as read" functionality for notifications.
5. Test notifications by flagging sample assignments.

[3.3] User Story 3: As an **admin**, I want to view trends of flagged assignments over time so that I can monitor patterns and take preventive action.

Tasks:

1. Fetch historical flagged assignment data.
2. Group the data by date to show weekly or monthly trends.
3. Create a line graph to visualize the trend.
4. Add the graph to the admin dashboard.
5. Test the trend display with different datasets.

[3.4] User Story 4: As a **user**, I want to access all my notifications on a clean dashboard so that I can manage them effectively.

Tasks:

1. Create a "Notifications" section on the dashboard.
2. Display all notifications with timestamps in a list view.
3. Add filters for unread and read notifications.
4. Provide an option to delete notifications.
5. Test the notification dashboard with a variety of sample notifications.

[3.5] User Story 5: As a **teacher**, I want to export fraud detection reports in a simple format so that I can share them with others.

Tasks:

1. Add an "Export Report" button to the report section.
2. Provide options to select export formats (e.g., PDF or CSV).
3. Format the report content for readability in the selected format.
4. Implement the functionality to download the report.
5. Test the export functionality with different report sizes and formats.

(4) Assignment Authenticity and Behavioural Insights

[4.1] User Story: As an **instructor**, I want a system that flags suspicious assignments so that I can easily review potential fraud cases.

Tasks:

1. Add a plagiarism detection algorithm using tools like Turnitin or custom methods.

2. Create a dashboard to display flagged assignments with a fraud score.

3. Save flagged submissions in a database with details like fraud score and reasons.

4. Integrate the detection system with assignment uploads.

5. Test the system with sample assignments for accurate detection.

[4.2] User Story: As a **student**, I want to know if my submission is flagged so that I can fix issues and resubmit.

Tasks:

1. Notify students when their submission is flagged.

2. Show reasons for flagging, like similarity or unusual patterns.

3. Provide an option to resubmit flagged assignments with tracking.

4. Provide anonymized examples or guidelines to help students understand why their submission was flagged.

5. Add a help section explaining the flagging process.

[4.3] User Story: As an **administrator**, I want to track system performance to ensure it works smoothly.

Tasks:

1. Build a dashboard to monitor system performance and detection rates.

2. Log flagged submissions and system activity for reviews.

3. Allow administrators to adjust detection thresholds as needed.

4. Generate automated reports on system performance and flagged cases.

5. Set up alerts for system errors or downtime.

[4.4] User Story: As an **instructor**, I want the system to detect collusion between students so that I can maintain academic integrity.

Tasks:

1. Compare assignments to detect similarity between multiple students.

2. Use text-matching algorithms to identify collusion.

3. Add visualizations to show patterns of suspicious behavior.

4. Test the system to ensure accurate detection of collusion.

5. Let instructors review and override flagged collusion cases.

[4.5] User Story: As a **student**, I want to ensure my data is private so that I feel secure using the system.

Tasks:

1. Encrypt all stored submissions to protect privacy.

2. Display clear privacy policies and terms of use for students.

3. Restrict access to flagged data to authorized users only.

4. Perform regular security audits to ensure data safety.

5. Allow students to request deletion of their old submissions.

(5) **Data Visualisation**

[5.1] As a **teacher**, I want to view an interactive heatmap showing trends of suspected fraud across assignments and students, so that I can identify patters and take corrective actions.

**Tasks**:

(1) Retrieve fraud detection data and aggregate it by assignments and students.

(2) Design and implement an interactive heatmap using appropriate charting tools.

(3) Enable filtering based on assignment type, date, or student group.

(4) Add hover-over tooltips to display detailed information about fraud scores.

(5) Test the heatmap functionality for usability and accuracy.

[5.2] As a **school administrator**, I want to see a comparitive line chart of student performance versus detected fraud instances.

**Tasks**:

(1) Collect and preprocess data for student grades and fraud scores.

(2) Plot grades on one axis and fraud scores on the other using a dual-axis chart.

(3) Allow the user to toggle between individual students or class- wide views.

(4) Implement a date range filter for customizing the analysis period.

(5) Validate data accuracy and ensure chart responsiveness across devices.

[5.3] As a **teacher**, I want to predict the likelihood of fraud based on the assignment type, so that I can adjust the difficulty or structure of assignments to minimize potential cheating.

**Tasks**:

(1) Use historical fraud detection data to create a predictive model that estimates the likelihood of fraud for different types of assignments (e.g., multiple choice, essays, group work).

(2) Display the predicted fraud risk for upcoming assignments based on their type.

(3) Allow teachers to adjust assignment structures based on fraud risk predictions (e.g., changing question formats, adding unique identifiers).

(4) Provide feedback to teachers about how their past assignments correlated with fraud detection results.

(5) Test the predictive model for accuracy and effectiveness in identifying high-risk assignments.

[5.4] As a **student**, I want to view my personal fraud detection score across assignments, so that I can understand if there have been any instances of suspicious behavior or potential cheating on my part.

**Tasks**:

(1) Retrieve the fraud detection scores for each assignment.

(2) Display the scores in a clear, easy-to-understand format (e.g., a bar chart or scorecard).

(3) Allow the student to filter the view by assignment or date.

(4) Provide a brief explanation or tooltip on how the fraud detection system works and how scores are calculated.

(5) Ensure that the score is updated after each assignment and accurately reflects the student's actions.

[5.5] As a **student**, I want to submit an appeal for a flagged fraud detection result, so that I can challenge any false or mistaken fraud alerts that may have been applied to my assignments.

**Tasks**:

(1) Identify assignments that have been flagged by the fraud detection system.

(2) Provide a "Submit Appeal" button next to flagged assignments for students to submit an appeal.

(3) Allow students to add comments or explanations to support their appeal.

(4) Send a notification to the teacher or administrator when an appeal is submitted.

(5) Provide feedback to the student on the outcome of the appeal.