**Project Overview, Setup instructions, User guide and API documentation**

**B09 Bootcamp Management System**

This documentation provides an overview of the B09 Bootcamp Management System project, which including the setup instructions, a detailed user guide, and API documentation for each of the core functions. The system is designed to handle student registration, validation, course enrollment, scoring, and reporting, along with other administrative functions.

**Project Overview**

The Management System is a Python-based application that manages student-related operations such as registration, fee validation, course enrollment, scoring, and report generation. The application incorporates data encryption, unique ID generation, email, and date of birth validation to ensure data security and integrity.

**Key Features:**

* Student Registration: Register students with email, DOB validation, and a unique ID generator.
* Fee Validation: Verify and manage student fee payments.
* Course Enrollment: Enroll students in various courses.
* Scoring and Results: Score students and view their results.
* Progress Report: Generate comprehensive progress reports for students.
* Data Encryption: Secure sensitive data through encryption techniques.

**Setup Instructions:**

- Ensure that Python is installed on your system.

- Required Modules: The system relies on a few Python modules. Install them by running:

1. cryptography
2. email-validator
3. datetime
4. re # Import the regular expressions module
5. import uuid # Import the uuid module to generate unique identifiers

Encryption setup can be found [here.](https://tecvinsonab.sharepoint.com/:w:/r/sites/Stage2-PythonBackend-BEP09/Shared%20Documents/BEP09/BEP09_BootcampMgtSystem%20Project/EncryptionSetupInstructions.docx?d=w667e52c79fe442a79bdfaec70078d0f4&csf=1&web=1&e=bUJ65G)

**User Guide:**

Once the application starts, the main menu will be presented with options to perform various actions such as registering a student, enrolling in courses, viewing student information, and generating reports. Each option is tied to a specific function that performs the corresponding task.

Student Registration:

The user is prompted to input student details such as name, email, date of birth (DOB), etc.

Email and DOB are validated to ensure correctness. A unique student ID is generated for each registration.

Fee Validation:

The system checks whether the student has paid the required fees for the course. Fee records are updated based on the student's enrollment and payment status.

Course Enrollment:

After registration, students can enroll in available courses. The system maintains a list of courses, and students can choose which ones to enroll in.

Scoring Students:

Teachers or administrators can enter scores for students in the enrolled courses. The system validates the input and stores the scores securely.

Viewing Results:

Students or administrators or instructors or parents can view the scored results for each course. The results can be filtered by course, or student ID.

Progress Report Generation:

A detailed report of the student's progress, including enrolled courses, scores, and overall performance, is generated.

Data Encryption:

Sensitive data such as student IDs and scores are encrypted for security purposes. The system ensures that only authorized users can access the decrypted data.

**API Documentation:**

The API Initialize necessary data structures and import required modules. Here we generally describe the guidelines and instructions on how to use an Application Programming Interface (API). Below is an overview of the typical sections included in API documentation:

1) Email, DOB Validation and Unique ID Generator. Purpose is to validate email and DOB, generate a unique student ID.

2) Student Registration Function. Purpose is to register a new student by validating the email and DOB, then saving the student information.

3) Data Encryption Function. Purpose is to encrypt and decrypt sensitive student data.

4) Fee Validation Function. Purpose is to validate student fee payments based on course enrollment.

5) Course Enrollment Function. Purpose is to enroll a student in a course if they are not already enrolled.

6) View Student Information Function. Purpose is to retrieve and display student information based on the student ID.

7) Scoring Students Courses Function. Purpose is to record scores for students in their enrolled courses.

8) Viewing Scored Results Function. Purpose is to view the scored results for a student.

9) Generate Student Progress Report. Purpose is to generate a detailed progress report for a student, which can be exported in various formats.

10) Main Menu Function. Purpose is to provide the main menu interface for users to interact with the system.

**Conclusion:**

This Management System is a comprehensive solution for handling student registration, course enrollment, scoring, and reporting. The system is secure with encryption features and ensures data integrity with validation mechanisms. Follow the setup instructions to get started, and use the user guide and API documentation to understand the system's functionalities.

**Technical Report**

**Development Process:**

The development of the BEP09 Bootcamp Manager system was structured into four key phases: requirements gathering and system design, core functionality development, testing and debugging, and deployment.

Phase 1: Requirements Gathering and System Design

During this phase, the team collaborated with stakeholders to identify the key requirements and functionalities needed for the system. This included understanding the needs of bootcamp administrators, instructors, and students. The system design was carefully planned to ensure scalability and security, incorporating features such as Role-Based Access Control (RBAC) and data encryption.

Phase 2: Core Functionality Development

In this phase, the core functionalities of the system were developed using Python. This included creating features for student registration, course enrollment management, and data validation. The development team focused on building a robust backend that could handle the dynamic nature of bootcamp environments.

Phase 3: Testing and Debugging

The testing phase involved rigorous testing using PyTest to ensure that all functionalities worked as intended. This included testing for edge cases, data validation, and security vulnerabilities. The team identified and resolved bugs, ensuring that the system was stable and secure before deployment.

Phase 4: Deployment

Finally, the system was deployed after thorough testing and validation. The deployment process included setting up the necessary infrastructure to host the system and ensuring that all security measures were in place.

Challenges Encountered and Solutions

**Challenges:**

Ensuring Data Security

Problem: One of the significant challenges was implementing robust security measures to protect sensitive student information. The system needed to ensure that data was secure both at rest and in transit, which required careful planning and implementation.

Solution: The team implemented Role-Based Access Control (RBAC) to restrict access to sensitive data based on user roles. Additionally, data encryption was employed to secure student information, ensuring that even if data were intercepted, it would remain unreadable and secure.

Managing Dynamic Course Enrollments

Problem: Another challenge was handling the dynamic nature of course enrollments in bootcamps, where students often change courses or enroll in new ones frequently. This required the system to be flexible and able to handle frequent updates without compromising data integrity.

Solution: The development team designed the system to allow administrators to easily update student information and manage course enrollments. This included features to add or remove courses for students and track enrollment numbers, ensuring that the system could adapt to the fast-paced bootcamp environment.

Automated Progress Tracking

Problem: Implementing automated progress tracking and reporting was a challenge, as it required the system to capture and analyze various data points related to student performance and course engagement.

Solution: The team integrated additional fields to track progress-related data and used Python’s reporting libraries to generate automated reports. This provided administrators with actionable insights into student performance without the need for manual tracking, enhancing the system's efficiency.

**Conclusion**

The development of the BEP09 Bootcamp Manager was a collaborative effort that involved careful planning, rigorous testing, and strategic problem-solving. Despite the challenges encountered, the team successfully delivered a secure, flexible, and scalable system that meets the unique needs of bootcamp environments. The project’s success is attributed to the team’s dedication to addressing challenges head-on and implementing solutions that enhanced the system's functionality and security.