Software Requirements and Design Document

For

Group 14

Version 1.0

Authors:
Samantha Bui
Ludginie Dorval
Antonio Garriga
William Lee
Lillian Malik

1. Overview (5 points)

ManuFactor is a web-based software program designed to simplify key accounting tasks for manufacturing businesses. The platform automates processes such as Cost-Volume-Profit (CVP) analysis and production budgeting, making it easier for companies to plan production, set sales targets, and achieve income goals. The system features three distinct user roles to control access and functionality: Admins, who can manage both products and user data; Users, who can manage products but have restricted access to user data; and Guests, who can only view product information. This role-based structure ensures the appropriate level of access for different types of users while maintaining a secure and efficient system.

2. Functional Requirements (10 points)

1. **User Management (High)**
   - **Admin Role**:
     - Ability to create, update, and delete user accounts.
     - As of Increment I, we intend to have list/add user.
   - **User Role**:
     - Can view their assigned product data but cannot modify user-related data or access other users' information.
     - As of Increment I, we intend to no user management.
   - **Guest Role**:
     - Can view publicly available product data, but has no ability to interact with or modify any data.
     - As of Increment I, we intend to no user management.
2. **Product Management (High)**
   - **Admin Role**:
     - Ability to add, update, and delete product data (such as product names, descriptions, and prices).
     - As of Increment I, we intend to have list/add data.
   - **User Role**:
     - Ability to add, update, and delete product information.
     - Users can view product data that is assigned to them but cannot access products that belong to other users. As of Increment I, we intend to have list/add data.
   - **Guest Role**:
     - Can only view a list of products with basic details such as name, description, and price, but cannot modify or add new products.
     - As of Increment I, we intend to have list data.

3. **Cost-Volume-Profit (CVP) Analysis (High)**
   - Ability to input data such as fixed costs, variable costs, and sales price for each product.
   - Ablility to generate CVP analysis reports based on user-provided data
   - Admins and Users can view the generated CVP reports.
   - Admins can generate CVP analysis reports for all products in the system.

4. **Production Budgeting (High)**
   - Users should be able to input production costs, expected units, and sales forecasts for budgeting purposes.
   - Ability to generate production budget reports that show projected costs, revenue, and profit margins.
   - Admins can view and manage budgets for all users, while Users can only view or manage their assigned products' budgets.

…

5. **Dashboard (Medium)**
   - Provide a dashboard that displays metrics for the user, such as:
     - **For Admin**: Overview of all users, products, and financial data.

    - **For User**: Overview of the products and related financial data.
    - **For Guest**: Overview of available products and basic product details.

6. **Access Control and Permissions (Medium)**
    - Implement role-based access control to ensure that users only have access to data and features appropriate for their role.
    - Admins can view and modify all data, while Users are restricted to managing only their assigned products and viewing certain reports.
    - Guests have read-only access to product information and cannot view any financial reports or make changes to any data.

7. **Security and Authentication (Medium)**
    - Implement secure login functionality using username and password.
        - Potentially including email as well.

8. **Data Storage and Backup (Medium)**
    - The system should store all user, product, financial data, and reports securely in a database.

9. **User Interface and Experience (Low)**
    - The application should have a clean, intuitive, and responsive user interface.
    - The interface should adapt for different devices (desktop, tablet, mobile) for easy access and use.
    - Provide easy navigation between user profiles, product management, financial reports, and system settings.

3. Non-functional Requirements (10 points)

1. **Performance**
    - **Response Time**: The system should load within 3 seconds for most operations (e.g., navigating between pages, generating reports).
    - **Throughput**: The system should be able to handle up to 20 concurrent users without performance degradation.

2. **Scalability**
    - The system should be scalable to accommodate future growth in the number of users and products without major changes to its structure.
    - It should be able to support an increasing number of products, users, and financial data.

3. **Security**
    - **Data Encryption**: All sensitive data should be encrypted.
    - **Authentication**: Implement role-based access control to ensure only authorized users have access to specific features and data.
    - **Authorization**: Ensure that users can only access and modify data according to their assigned roles (Admins, Users, and Guests).
    - **Data Privacy**: The system must ensure user data is protected.

…

4. **Usability**
    - **User Interface (UI)**: The system must be easy to navigate, with a clean and intuitive design, so users can quickly find and use features.

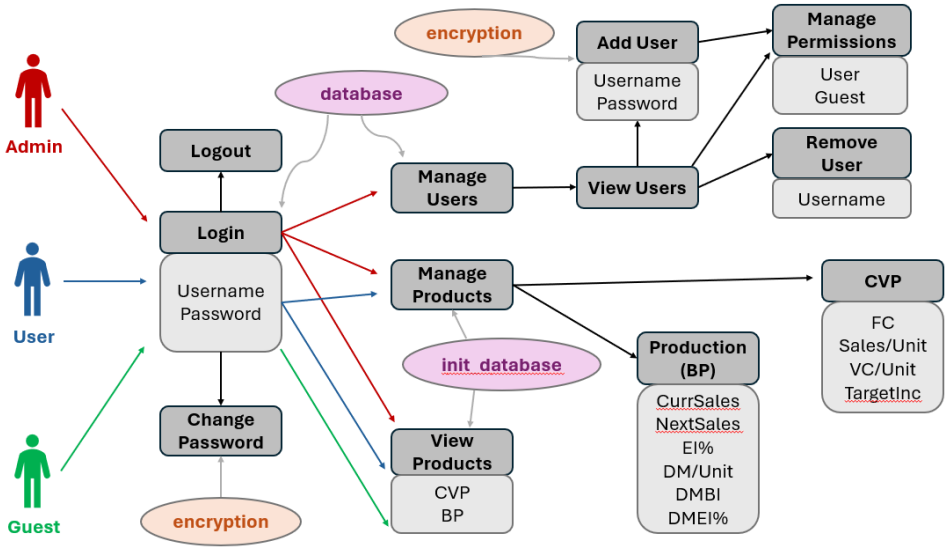
- **Responsive Design**: The application should be fully responsive, ensuring it works on various devices, including desktops, tablets, and smartphones.

5. **Maintainability**
   - **Code Quality**: The codebase should follow best practices for clean, modular, and well-documented code to allow for easy updates for future increments.
   - **Testing**: Comprehensive tests should be implemented to ensure reliability and quality.

6. **Modularity**
   - **Extensibility**: The system should be designed with extensibility in mind, so new features can be easily added.

7. **Compatibility**
   - **Cross-Browser Compatibility**: The system should be fully compatible with the latest versions of major web browsers.
   - **Operating System Compatibility**: The system should be compatible with commonly used operating systems.

4. Use Case Diagram (10 points)

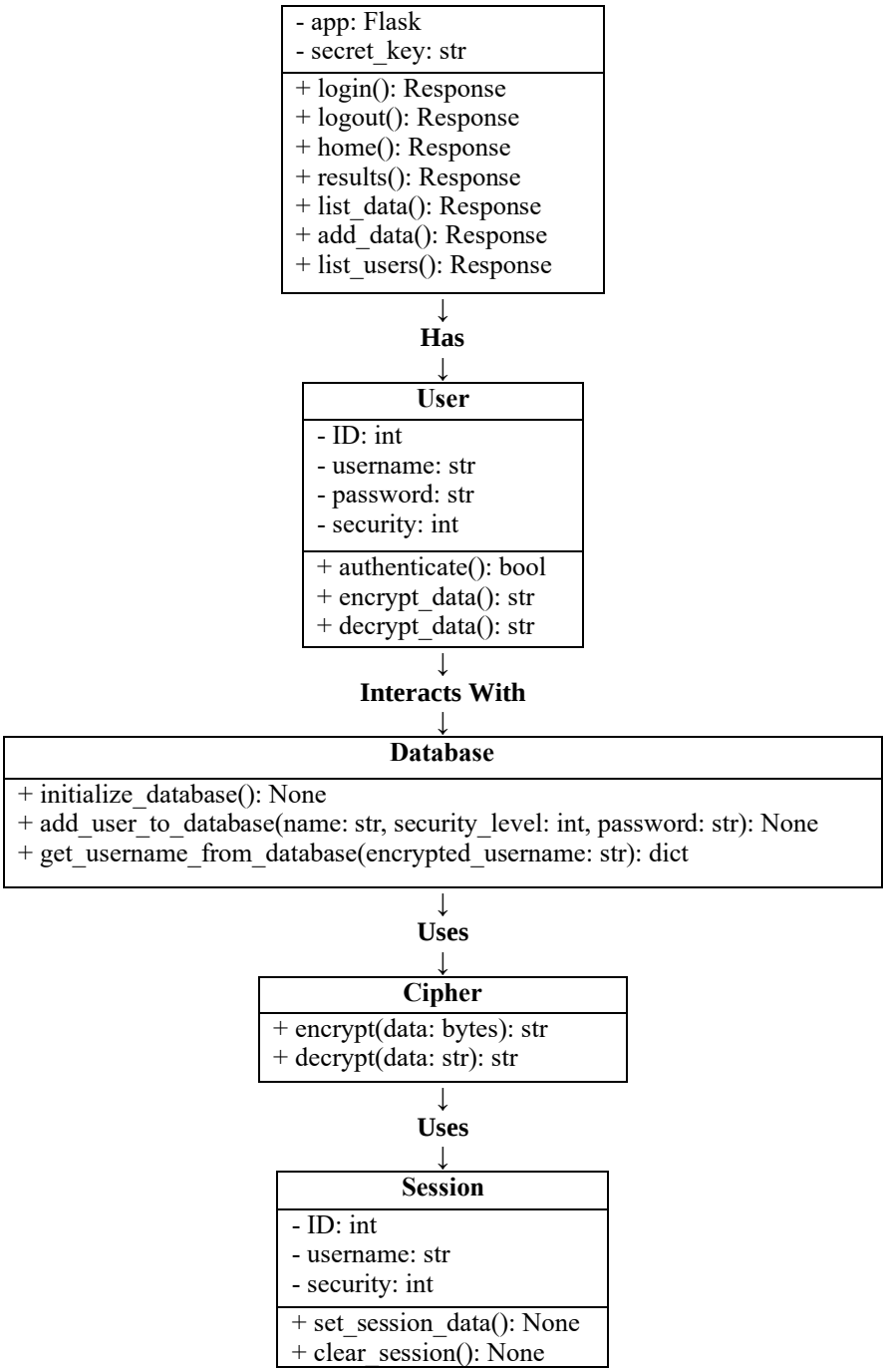


Please be advised that the Production (BP) feature has not yet been implemented. It is scheduled for inclusion in Increment 2, along with several other additional features.

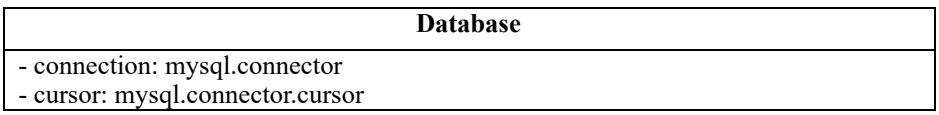
5. Class Diagram and/or Sequence Diagrams (15 points)

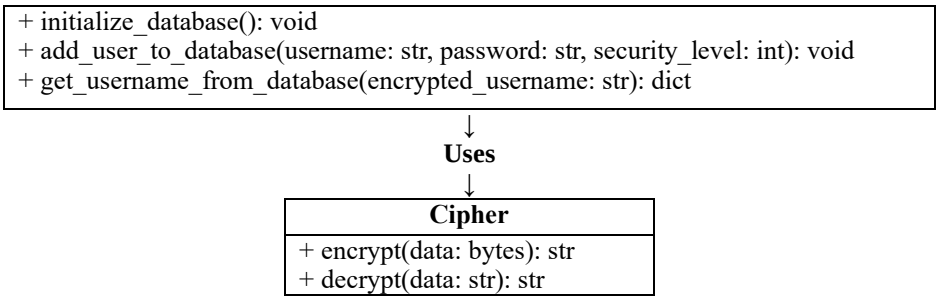
FRONT-END CLASS DIAGRAM (*app.py*)

Flask App

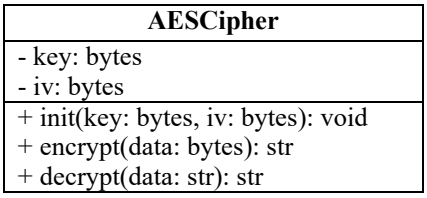


FRONT-END DATABSE CLASS DIAGRAM (*database.py*)

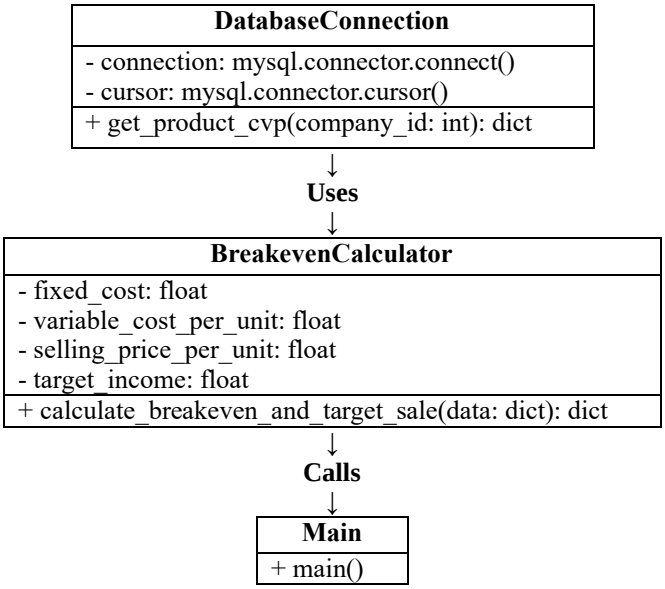




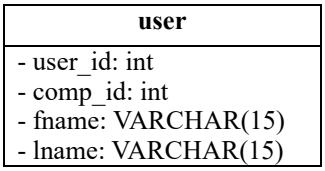
ENCRYPTION CLASS DIAGRAM (*encryption.py*)



BACK-END CLASS DIAGRAM (*product_cvp.py*)



BACK-END DATABSE CLASS DIAGRAM (*init_database.sql*)



- email: VARCHAR(20)

product_cvp
- prod_id: int - comp_id: int - fixed_cost: DECIMAL(15,2) - variable_cost_per_unit: DECIMAL(10,2) - selling_price_per_unit: DECIMAL(10,2) - target_income: DECIMAL(15,2)

ADMIN SEQUENCE DIAGRAM:

Adding Users/Managing Products

Login
Admin -> Flask App: login() Flask App -> User: authenticate(username, password) User -> Database: get_username_from_database(encrypted_username) Database -> Cipher: decrypt(data) Cipher -> Database: return decrypted data Database -> User: return authentication result User -> Flask App: return authentication result Flask App -> Session: set_session_data() Flask App -> Admin: redirect to home()

View User Data
Admin -> Flask App: list_users() Flask App -> Database: get_users() Database -> Cipher: decrypt(user_data) Cipher -> Database: return decrypted data Database -> Flask App: return user data Flask App -> Admin: return user data (list)

Add New User
Admin -> Flask App: add_data(name, security_level, password) Flask App -> Database: add_user_to_database(name, security_level, encrypted_password) Database -> Cipher: encrypt(password) Cipher -> Database: return encrypted password Database -> Flask App: return success Flask App -> Admin: confirm user added

…

Manage Products
Admin -> Flask App: list_data() Flask App -> Database: get_product_cvp(company_id) Database -> Flask App: return product data Flask App -> BreakevenCalculator: calculate_breakeven_and_target_sale(data) BreakevenCalculator -> Flask App: return CVP analysis results Flask App -> Admin: display CVP analysis results

USER SEQUENCE DIAGRAM:

Viewing Data/CVP Analysis

Login
User -> Flask App: login() Flask App -> User: authenticate(username, password) User -> Database: get_username_from_database(encrypted_username) Database -> Cipher: decrypt(data) Cipher -> Database: return decrypted data Database -> User: return authentication result User -> Flask App: return authentication result Flask App -> Session: set_session_data() Flask App -> User: redirect to home()

View Product Data
User -> Flask App: list_data() Flask App -> Database: get_product_cvp(company_id) Database -> Flask App: return product data Flask App -> User: return product data (list)

Perform CVP Analysis
User -> Flask App: results() Flask App -> Database: get_product_cvp(company_id) Database -> Flask App: return product data Flask App -> BreakevenCalculator: calculate_breakeven_and_target_sale(data) BreakevenCalculator -> Flask App: return CVP analysis results Flask App -> User: display CVP analysis results

GUEST SEQUENCE DIAGRAM:

Viewing Only

Login
Guest -> Flask App: login() Flask App -> User: authenticate(username, password) User -> Database: get_username_from_database(encrypted_username) Database -> Cipher: decrypt(data) Cipher -> Database: return decrypted data Database -> User: return authentication result User -> Flask App: return authentication result Flask App -> Session: set_session_data() Flask App -> Guest: redirect to home()

View Product Data
Guest -> Flask App: list_data() Flask App -> Database: get_product_cvp(company_id) Database -> Flask App: return product data Flask App -> Guest: return product data (view only)

…

View CVP Analysis Results
Guest -> Flask App: results() Flask App -> Database: get_product_cvp(company_id) Database -> Flask App: return product data Flask App -> BreakevenCalculator: calculate_breakeven_and_target_sale(data) BreakevenCalculator -> Flask App: return CVP analysis results Flask App -> Guest: display CVP analysis results

6. Operating Environment (5 points)

Hardware Platform:

- Users can access the system on desktops, laptops, tablets, and mobile devices, provided they have an internet connection.

Operating System:

- Since the software will be accessed through a web browser, it is compatible with most major operating systems.
  - Users will not need to install any specific operating system software for the platform, just a modern browser.

Web Browser Compatibility:

- The web application will support the latest versions of commonly used browsers, and older versions may not be compatible or cause issues for the user.

Software Components:

- **Frontend**:
  - The application will be built using HTML, CSS, and JavaScript.
  - The frontend will communicate with the backend using JSON for data exchange.
- **Backend**:
  - The backend of ManuFactor will be a Python-based server (using Flask) for efficient data handling and user requests.
  - The system will rely on an SQL-based database (using MySQL) to store user data, product information, and financial records.

7. Assumptions and Dependencies (5 points)

Assumptions:

- **User Access to Modern Browsers**: It is assumed that users of ManuFactor will have access to modern web browsers and will keep them up to date.
- **Stable Internet Connectivity**: The application is a web-based system, so it assumes stable internet access for users to interact with the software.
- **Sufficient Hardware for End Users**: It is assumed that end users will be using devices that are capable of running modern web browsers.
- **If any of these assumptions are incorrect, the Webpage cannot be reached**.

Dependencies:

- **Database Management Systems**: The project assumes that the SQL-based database (MySQL) will be stable and perform well. Any issues in the database management system could impact performance.
- **Security Tools**: The project will depend on services for encryption (Cipher). Any issues regarding Cipher could pose security risks.

…

Constraints:

- **Development Timeline**: The project timeline is relatively short, which could affect the project's quality.