SDLC Planning - The Count of Money

Part 1: Identifying Resources and Estimating Time

1. Required Security Tools

1.1 Dependency Analysis

- **npm audit**: Analyzes vulnerabilities in Node.js packages
- **Snyk**: Continuous dependency scanning
- **GitHub Security Alerts**: Alerts for vulnerable dependencies

1.2 Security Testing

- **OWASP ZAP**: Dynamic security testing
- **Postman**: API testing
- **Trivy**: Docker container analysis

1.3 Authentication and Authorization

- **JWT**: Authentication token management
- **OAuth2**: Google authentication
- **bcrypt**: Password hashing

1.4 Frontend Security

- **Validator.js**: Input validation
- **DOMPurify**: XSS protection
- **Vue Router Guards**: Route protection

1.5 API Security

- **Helmet.js**: Security headers
- **express-rate-limit**: Brute-force attack protection
- **CORS**: Secure configuration

2. Git Structure

See **GitHub**.

3. Detailed Planning

See **Jira**.

4.Resource Allocation by Component

Authentication

- JWT Decoder
- Password Validator
- OAuth2 Client
- Session Manager

Routes and Navigation

- Route Guards
- Permission Manager
- Error Handler
- Security Middleware

API and Data

- Input Validator
- Rate Limiter
- CORS Manager
- Data Sanitizer

Secure UI/UX

- XSS Protection
- CSP Configuration
- Error Boundary
- Session Monitor

5.Total Estimated Time: 25 Business Days

- Configuration and Setup: 5 days
- Security Development: 17 days
- Testing and Audit: 3 days

Documenting Security Measures

1. Identifying Sensitive Features

Our application contains several features that require special security attention.

Authentication Module

The authentication system is critical as it serves as the entry point to the application. It manages:

- Email/password login
- Google OAuth authentication
- New user registration
- User session management

This functionality is particularly sensitive as a vulnerability could compromise the entire system and user data.

Anonymous Interface

Even though the anonymous interface is public, it requires protection as it:

- Displays the **N** most popular cryptocurrencies
- Presents the **K** latest articles
- Manages trend and percentage displays

Security measures are necessary to prevent data manipulation and maintain the integrity of displayed information.

User Interface

This interface handles personalized data:

- Personal cryptocurrency list
- Keywords for news review
- Profile preferences
- Modification history

Protecting these features is essential to ensure user data confidentiality.

Administrator Interface

The administrator interface is the most sensitive as it allows:

- Managing global preferences
- Configuring available cryptocurrencies
- Managing article sources
- Adjusting general system settings

2. Critical Processes

Identified critical processes include:

Identity Management

- Account creation and validation
- Session management
- Profile modification
- Password reset

Financial Data Management

- Real-time exchange rate retrieval
- Storing user preferences
- Data history tracking
- Information synchronization

Content Management

- Article aggregation
- Keyword filtering
- Source validation
- Content distribution

3. Assigning Responsibilities (RACI)

Process	Responsible (R)	Accountable (A)	Consulted (C)	Informed (I)
Authentication Service	Authentication Service, Backend Developers, Token Management Service	Security Administrator, Project Manager	User Service, Security Expert, OAuth Service	End Users, Technical Support, Monitoring Team
Cryptocurrency Management	Crypto Data Service, API Developers, Update Service	System Administrator, Data Manager	External APIs, Financial Expert, Validation Service	Logged-in Users, Monitoring Service, Support Team
Article Management	Article Aggregation Service, Backend	Content Administrator,	RSS Parser, Content	All Users, Support Team,

Process	Responsible (R)	Accountable (A)	Consulted (C)	Informed (I)
	Developers, Filtering Service	Project Manager	Expert, Validation Service	Monitoring Service
User Management	User Management Service, Frontend/Backend Developers, Profile Service	System Administrator, GDPR Officer	Legal Service, Security Expert, Validation Service	Concerned Users, Technical Support, Monitoring Service
Administration	Administration Service, System Developers, Configuration Service	Principal Administrator, Technical Management	Security Expert, Technical Service, System Auditors	All Administrators, Development Team, Monitoring Service

4. Converting RACI Roles to RBAC (Role-Based Access Control)

To facilitate the implementation and visualization of the access control system, the following tables summarize the RBAC structure:

RBAC Overview Table

Role	Access Level	Scope	Action Zone
System Administrator	Full	Global	All functionalities
Content Moderator	Partial	Content	Articles and sources
Authenticated User	Restricted	Personal	Profile and preferences
Anonymous User	Minimal	Public	Read-only access

Detailed Permissions Matrix

Feature	Admin	Moderator	User	Anonymous	Justification
User Management	CRUD	-	Read/Update (self)	-	Security and maintenance
System Configuration	CRUD	-	-	-	System integrity

Feature	Admin	Moderator	User	Anonymous	Justification
Article Management	CRUD	CRUD	Read	Read	Content quality
Cryptocurrency Management	CRUD	Read	Read/Save	Read	Core service
Profile Settings	CRUD	Read	CRUD (self)	-	User control
RSS Sources	CRUD	CRUD	-	-	Content feed
Statistics	CRUD	Read	Read	Read	Transparency
System Logs	CRUD	-	-	-	Audit and security

Sensitive Actions and Validation

Action	Required Validation	Authorization Level	Log
Account Deletion	Two-factor authentication	Admin	Yes
Role Modification	Two-factor authentication	Admin	Yes
Adding RSS Source	Simple authentication	Moderator	Yes
Cryptocurrency Modification	Two-factor authentication	Admin	Yes
Password Reset	Simple authentication	Admin/Self	Yes

This table-based structuring complements the narrative description of the RBAC system and provides a clear reference for implementing access controls in the application.

5. Security Flow Integration into Workflow

To ensure optimal security throughout the application's lifecycle, we have integrated security mechanisms into each step of the user workflows. This approach ensures continuous protection without compromising user experience.

Application Access Workflow

1. Authentication

- **Entry Point**: Login interface
- **Integrated Security Measures**:
  - Real-time input validation
  - Password complexity verification
  - Brute-force attack protection (rate limiting)
  - Two-factor authentication for sensitive actions
- **Secure Flow**:
  - Login attempt → Credentials validation → 2FA verification (if enabled) → JWT token generation → Access granted

2. Navigation and Consultation

- **Entry Point**: Main interface
- **Integrated Security Measures**:
  - Token authenticity verification for each request
  - XSS protection in content display
  - CSRF tokens for user actions
  - Secure loading of external resources
- **Secure Flow**:
  - Page request → Token verification → Permission validation → Secure display

Data Management Workflow

1. Cryptocurrency Handling

- **Interaction Points:** Price interface and custom lists
- **Integrated Security Measures:**
  - Data integrity verification
  - Protection against API manipulation
  - Rate limiting on data requests
  - Data validation before display
- **Secure Flow:**
  - Price request → API authentication → Data validation → Secure display → Validated user actions

2. Article Management

- **Interaction Points:** Article interface and filters
- **Integrated Security Measures:**
  - RSS source validation
  - HTML content sanitization
  - Access control on restricted articles
  - Protection against injection in search filters
- **Secure Flow:**
  - Article retrieval → Source validation → Sanitization → Filtering → Secure display

Administrative Workflow

1.System Management

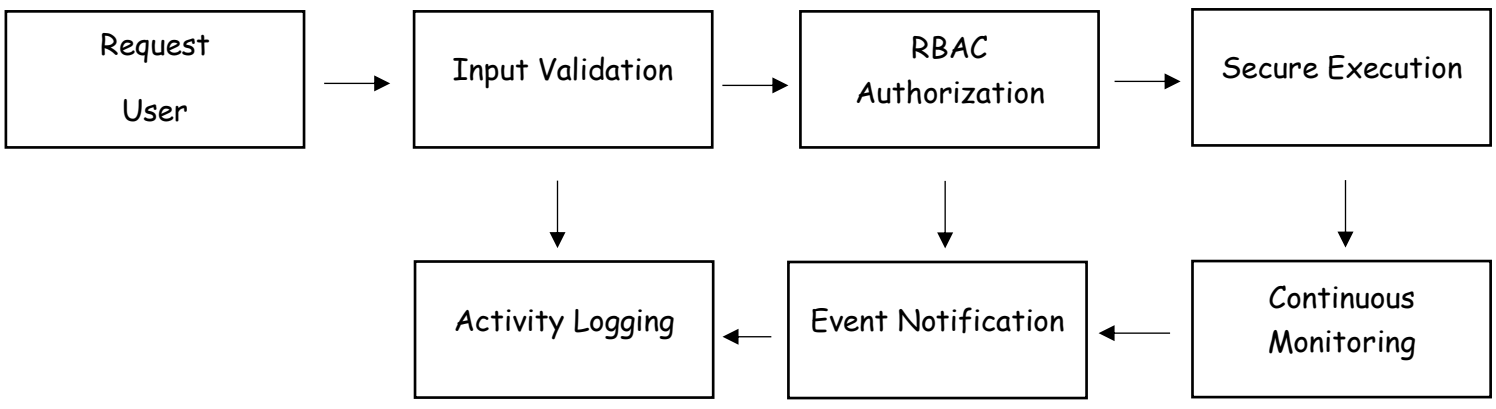
- **Interaction Points:** Administration interface
- **Integrated Security Measures:**
    - Mandatory two-factor authentication
    - Detailed action logging
    - Privilege verification for each action
    - Explicit confirmation for critical actions
- **Secure Flow:**
    - Admin access → 2FA → Privilege verification → Action → Confirmation → Logging

2.User Management

- **Interaction Points:** Admin panel
- **Integrated Security Measures:**
    - Privilege separation
    - Multi-step validation for sensitive actions
    - Notification to affected users
    - Complete audit trail
- **Secure Flow:**
    - Action request → Privilege validation → Confirmation → Execution → Notification → Logging

Secure Workflow Diagram

Diagramme de Workflow Sécurisé



6. Integration of Security Flow into the Workflow

The security analysis allowed us to identify the most critical features that require priority attention during the implementation of security measures. This prioritization is essential for efficiently allocating resources and ensuring that the most sensitive elements are protected first.

Prioritization Criteria

We evaluated each feature based on the following criteria:

- **Impact:** Potential consequences of a compromise (1-5)
- **Probability:** Likelihood of a successful attack (1-5)
- **Exposure:** Level of accessibility of the feature (1-5)
- **Dependencies:** Number of systems affected in the event of a compromise (1-5)

The criticality score is calculated using the formula: (Impact × 2) + Probability + Exposure + Dependencies

Feature Prioritization Table

Rang	Feature	Impact	Probability	Exposure	Dependencies	Score	Priority
1	Authentication	5	5	5	5	25	Critical
2	Admin Panel	5	4	3	5	22	Critical
3	Crypto API Management	4	4	4	4	20	High
4	User Management	4	3	4	4	19	High
5	Sensitive Data Storage	5	3	2	3	18	High
6	RSS Validation	3	4	3	3	16	Medium
7	User Forms	3	4	4	2	16	Medium
8	Search System	2	3	5	2	14	Medium
9	Article Display	2	3	5	1	13	Medium

Rang	Feature	Impact	Probability	Exposure	Dependencies	Score	Priority
10	Profile Preferences	2	2	3	2	11	Low
11	Public Statistics	1	2	5	1	10	Low
12	Anonymous Interface	1	2	5	1	10	Low

Justification of Critical Priorities

1. **Authentication (Score: 25 - Critical)**

   Authentication is the main entry point of the application. A compromise at this level would allow unauthorized access to the entire system and user data. Its critical priority is justified by:

   - Maximum impact in case of a breach
   - Continuous exposure to attack attempts
   - Critical dependencies with all other systems

2. **Admin Panel (Score: 22 - Critical)**

   The admin panel provides access to the most sensitive features of the system. Its critical priority is justified by:

   - Full control over the application
   - Access to all user data
   - Ability to modify system settings

3. **Crypto API Management (Score: 20 - High)**

   Cryptocurrency APIs are at the core of the application's functionality. Their high priority is justified by:

   - Possible manipulation of financial data
   - Continuous exposure to external queries
   - Direct impact on the service's reliability

Secure Implementation Plan

1. **Phase 1 - Critical Priority (Weeks 1-2)**

- Implement security measures for authentication
- Secure the admin panel
- Protect role management systems

2. **Phase 2 - High Priority (Weeks 3-4)**

    - Secure cryptocurrency APIs
    - Protect user data
    - Implement secure storage

3. **Phase 3 - Medium Priority (Weeks 5-6)**

    - Secure forms and validations
    - Protect search systems
    - Secure RSS flows

4. **Phase 4 - Low Priority (Weeks 7-8)**

    - Secure public interfaces
    - Protect statistics
    - Finalize access controls

This prioritization allows for a structured approach to implementing security measures, starting with the most critical elements to ensure effective protection of the system's essential features from the start of development.

Final Document of Sensitive Features

This document summarizes all the sensitive features of the "The Count of Money" application and details the required security measures for each.

1. **Authentication System**

    Description: Main entry point of the application, managing user identification via email/password and OAuth2.

    Required Security Measures:

    - Secure password hashing with bcrypt (cost factor ≥ 12)
    - Strict password policy (minimum 12 characters, complexity)
    - Rate limiting (max 5 attempts/minute) to prevent brute-force attacks

- JWT tokens with short expiration (60 minutes) and refresh token rotation
- Two-factor authentication for sensitive actions
- Full validation of OAuth2 tokens with domain verification
- Automatic logout after 30 minutes of inactivity
- Logging of suspicious logins (IP, timestamps, User-Agent)

2. Admin Panel

Description: Interface reserved for administrators to manage the application globally.

Required Security Measures:

• Strong authentication (2FA required)
• Access limited to authorized IP addresses
• Full auditing of all administrative actions
• Privilege separation for critical actions
• Reduced session expiration time (15 minutes)
• Explicit confirmation for any critical modification
• Real-time notifications of system changes
• Non-predictable and non-publicly exposed access routes

3. Cryptocurrency API

Description: Services managing retrieval, storage, and display of cryptocurrency data.

Required Security Measures:

- API authentication via tokens with specific permissions
- Data validation at both source and reception
- Rate limiting (100 requests/minute per user)
- Protection against denial-of-service attacks
- Encryption of sensitive data
- Data integrity verification
- Continuous monitoring of suspicious access patterns
- Secure caching to limit external queries

4. User Management

Description: Features for creating, modifying, and deleting user accounts.

Required Security Measures:

- Strict validation of user inputs
- Sanitization of data before storage
- Granular permissions based on roles
- Protection against account enumeration
- Secure password recovery process
- Notification of account modifications
- Logging of sensitive changes
- Email address verification

5. Personal Data Storage

Description: Mechanisms for storing and managing users' personal information.

Required Security Measures:

- Encryption of sensitive data at rest
- Isolation of data by user
- Retention and automatic deletion policies
- Full GDPR compliance
- Access limited to the principle of least privilege
- Encrypted backups with rotation
- Pseudonymization of non-essential data
- Audit trail of access to personal data

6. RSS Feed and Article System

Description: Features for aggregation, filtering, and displaying news articles.

Required Security Measures:

- Validation of authorized RSS sources
- of HTML content before display
- Protection against XSS attacks
- Filtering of malicious content
- Integrity verification of feeds
- Rate limiting on aggregation requests
- Isolation of parsing processes
- Secure keyword-based filtering

7. User Forms

Description: All forms for data entry and submission by users.

Required Security Measures:

- Server-side validation of all inputs
- CSRF protection on every form
- Sanitization of incoming data
- Filtering of special characters
- Recaptcha for public forms
- Limited error feedback (no information disclosure)
- Data type validation

8. Search System

Description: Features for searching articles and cryptocurrencies.

Required Security Measures:

- Protection against SQL/NoSQL injections
- Validation and sanitization of search terms
- limiting (20 searches/minute)
- Limitation of results per page
- Protection against denial-of-service attacks
- Secure indexing
- Restriction of sensitive search fields
- Logging of unusual search patterns

9. Profile Preferences

Description: Features allowing users to personalize their experience.

Required Security Measures:

- Validation of preference changes
- Protection against preference manipulation
- Limitation of customizable options
- Verification of data consistency
- Notification of significant changes

- Secure restoration of default settings
- Audit of modifications
- Protection against malicious preferences

10. Public Interface

Description: Features accessible without authentication.

Required Security Measures:

- Limitation of exposed data
- Protection against scraping
- Rate limiting by IP
- DDoS protection
- Sanitization of displayed data
- Secure caching
- HTTP security headers
- Strict Content Security Policy

This document will serve as a reference for the implementation and verification of security measures throughout the development cycle, ensuring each sensitive feature is appropriately protected.

Part 2: Design and Modeling

MongoDB Database Analysis

The examination of the MongoDB database was conducted using MongoDB Shell, allowing direct access to the collections and their structures. This analysis revealed that the "The Count of Money" application relies on two main collections: "users" and "anonyms." The "users" collection stores information related to registered users, including personal data, login credentials, and preferences. Analyzing a sample document revealed the following structure:

```
{
_id: ObjectId('67b8e31624a8db97bda17e73'),
avatar: null,
lastname: "debede",
firstname: 'sole',
username: 'debede',
email: 'debedenouwo@gmail.com',
password: '$2b$10$Tm64j4LOxePHLlXDwpTBteVJuvO0hScIeYdhb9QazH5V9QLRXVvi6',
role: 'user',
articles: [],
cryptos: [],
__v: 0
}
```

The "anonyms" collection contains the default settings for unauthenticated users, with the following structure:

```
{
 _id: ObjectId('67b3cbab61fd6ac0cf2175d1'),
 name: 'Default',
 NombreArticles: 3,
 NombreCryptos: 3,
 articles: [],
 cryptos: [],
 __v: 0
}
```

Identification of Security Risks

A thorough analysis of the database identified several potential vulnerabilities. To achieve this, various test commands were executed to assess the system's robustness:

```
// Test des vulnérabilités d'injection
db.users.find({username: {$regex: /<script>/}})

// Vérification des contrôles d'accès
db.getUsers()
// Résultat: { users: [], ok: 1 }

// Tentative de vérification du schéma
db.users.getSchema()
// Erreur: TypeError: db.users.getSchema is not a function
```

These tests revealed several significant security issues:

1. **Absence of Access Controls**: The absence of configured MongoDB users (as demonstrated by the db.getUsers() command) indicates that the database lacks dedicated authentication mechanisms, creating a risk of unauthorized access in case of server compromise.
2. **Lack of Schema Validation**: The absence of schema validators exposes the database to the insertion of malformed or malicious data, compromising data integrity.
3. **Unsecured Storage of Sensitive Data**: Personal information such as email addresses and names are stored in plaintext, without encryption or pseudonymization mechanisms.
4. **Vulnerability to NoSQL Injections**: The application may be vulnerable to NoSQL injections if it uses dynamically constructed queries from unvalidated user inputs.

5. **Insufficient Indexing**: The lack of proper indexing could not only affect performance but also expose the application to denial of service attacks via resource-intensive queries.

Countermeasures for Identified Threats

Our threat modeling analysis highlighted several potential vulnerabilities in the architecture of the "The Count of Money" application. For each of these threats, we propose specific countermeasures aimed at strengthening the security of the system.

Privilege Escalation
Privilege escalation threats allow an attacker to access features requiring higher rights than those they legitimately possess. To counter these risks, we recommend:

- Implementing an RBAC (Role-Based Access Control) system in MongoDB to strictly limit access based on assigned roles.
- Setting up an authentication middleware in Express.js to systematically verify authorizations before granting access to protected routes.
- Configuring MongoDB users with limited privileges for common operations, thus avoiding the use of administrator accounts for routine application functions.

Information Disclosure
Information disclosure vulnerabilities expose sensitive data to unauthorized users. To protect this data:

- Implementing encryption of sensitive data at rest in MongoDB, particularly for user personal information.
- Enabling TLS/SSL encryption for all communications with MongoDB to prevent interception of data in transit.
- Using secure projections in MongoDB queries to exclude sensitive fields from results.
- Configuring HTTP security headers via Helmet.js to reduce the risks of exposure of sensitive information.

NoSQL Injections
NoSQL injection attacks allow manipulation of database queries. To protect against this:

- Implementing Mongoose with strict schema validation for each collection.
- Using express-validator to validate and sanitize all user inputs before processing.
- Avoiding dynamic query constructors or dangerous operators like $where, which can be exploited in injection attacks.

Repudiation
Repudiation issues arise when a user can deny performing an action due to a lack of traceability. To address this:

- Implementing a comprehensive logging system to record all critical user actions.
- Configuring an audit trail in MongoDB to trace important data modifications.
- Using cryptographic signatures for critical actions to ensure their non-repudiation.

Forgery
Forgery threats concern unauthorized modifications of data. To counter these risks:

- Implementing digital signatures to verify the integrity of sensitive data.
- Setting up strict MongoDB schema validators to prevent the insertion of incorrect data.
- Using Mongoose hooks to validate changes before they are saved to the database.

Denial of Service
Denial of service attacks aim to make the application unavailable. To protect against this:

- Implementing robust rate limiting to limit the number of requests per IP.
- Creating MongoDB indexes for frequent query performance optimization.
- Limiting the size of queries to prevent memory or network saturation attacks.
- Configuring appropriate timeouts for MongoDB operations to avoid infinite queries.

The systematic application of these countermeasures, along with continuous monitoring and regular security audits, will significantly reduce the risks identified during threat modeling and ensure an appropriate level of security for the "The Count of Money" application.

The rest of the answers to the questions can be found in the GitHub repository SDLC-project.