

Cybersecurity Bootcamp Unit 3 Project

Table of Contents

[Disclaimer 3](#_dfo2dl77lw2c)

[Company Profile 3](https://docs.google.com/document/d/17YIl6WHRzqy0yUjmQErKUA3UK6RGXh7N0-GP1PESCms/edit#heading=h.ansdpjqo6qrr)

Spendology Solutions [3](https://docs.google.com/document/d/17YIl6WHRzqy0yUjmQErKUA3UK6RGXh7N0-GP1PESCms/edit#heading=h.sy2t99yjqvfa)

SpendSmart Budgeting Application [3](https://docs.google.com/document/d/17YIl6WHRzqy0yUjmQErKUA3UK6RGXh7N0-GP1PESCms/edit#heading=h.sy2t99yjqvfa)

Technical Implementation Details [3](https://docs.google.com/document/d/17YIl6WHRzqy0yUjmQErKUA3UK6RGXh7N0-GP1PESCms/edit#heading=h.sy2t99yjqvfa)

Database Schema5

SDLC Process7

# 

Case Study

# Disclaimer

All entities and events in this case study are purely hypothetical. Details of the case study are inspired by real-world events.

# Company Profile: Spendology Solutions

Spendology Solutions is a fictional digital age start-up fintech company dedicated to revolutionizing the personal financial management experience. Established in 2017, the company is driven by a passion for delivering innovative financial solutions that empower users to take control of their personal finances. With a commitment to security, usability, and efficiency, Spendology Solutions aims to set new standards in the ever-evolving landscape of digital financial management.

Our mission at Spendology Solutions is to simplify and enhance the financial lives of individuals through the creation of intuitive, secure, and feature-rich digital banking applications. We strive to provide our users with tools that enable efficient budgeting, seamless data integration, and a trustworthy platform for managing their financial activities.

# SpendSmart Digital Budgeting App

The SpendSmart digital budgeting app is designed to empower customers with efficient monthly budgeting capabilities, consolidating data from multiple credit card statements and utilities companies. This feature aims to provide a seamless and secure experience for users to manage their financial activities, all within the app.

## Technical Implementation Details:

* Data Integration:
  + Fetch and integrate data from various external sources, including credit card and utilities companies, leveraging token-based authentication.
  + Utilize encryption protocols during data transmission to ensure the confidentiality and integrity of the transferred data.
* Data Parsing and Normalization:
  + Develop algorithms to parse and normalize data from diverse sources, ensuring consistency in formatting and structure.
  + Implement error handling mechanisms to address discrepancies and anomalies in the incoming data.
* User Authentication and Authorization:
  + Integrate multi factor authentication mechanisms to protect user portals, ensuring that only authorized users can access their financial data.
  + Implement role-based access controls for admins/staff engineers to restrict access to specific features and functionalities based on user roles.
* Secure Storage:
  + Employ industry-standard encryption algorithms for storing sensitive financial data in databases.
    - Encrypt sensitive fields in the database before storage using AES-256 encryption.
    - Employ proper key management practices to safeguard encryption keys.
  + Regularly audit and update encryption mechanisms to comply with evolving security standards.
* Transaction Logging:
  + Establish a secure logging mechanism to record user interactions and financial transactions for audit purposes.
    - Log relevant events, including user logins, financial transactions, and system activities, using a secure and standardized logging framework.
    - Include timestamp information, user identifiers, and transaction details in each log entry.
  + Ensure logs are protected against unauthorized access and regularly reviewed for potential security incidents.
    - Store logs in a secure location with restricted access, ensuring that only authorized personnel can view or modify the log files.
    - Implement access controls and encryption to protect logs from unauthorized tampering or disclosure.
* User Education and Communication:
  + Implement user-friendly interfaces and informative tooltips to guide users on secure financial practices.
  + Communicate security features to users, fostering a sense of trust in the application.

### 

## Database Schema:

`budget\_categories` (

`budget\_category\_id` int unsigned NOT NULL AUTO\_INCREMENT,

`user\_id` int unsigned DEFAULT NULL,

`category\_name` varchar(255) DEFAULT NULL,

`budget\_limit` decimal(10,2) DEFAULT NULL,

PRIMARY KEY (`budget\_category\_id`),

KEY `user\_id` (`user\_id`),

CONSTRAINT `budget\_categories\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`)

)

`credit\_card\_statements` (

`credit\_card\_statement\_id` int unsigned NOT NULL AUTO\_INCREMENT,

`user\_id` int unsigned DEFAULT NULL,

`credit\_card\_number` varchar(16) DEFAULT NULL,

`institution` varchar(255) DEFAULT NULL,

`transaction\_date` date DEFAULT NULL,

`merchant` varchar(255) DEFAULT NULL,

`amount` decimal(10,2) DEFAULT NULL,

PRIMARY KEY (`credit\_card\_statement\_id`),

KEY `user\_id` (`user\_id`),

CONSTRAINT `credit\_card\_statements\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`)

)

`user` (

`user\_id` int unsigned NOT NULL AUTO\_INCREMENT,

`username` varchar(255) DEFAULT NULL,

`password\_hash` varchar(255) DEFAULT NULL,

`email` varchar(255) DEFAULT NULL,

PRIMARY KEY (`user\_id`)

)

`user\_expenses` (

`user\_expense\_id` int unsigned NOT NULL AUTO\_INCREMENT,

`user\_id` int unsigned DEFAULT NULL,

`category\_id` int unsigned DEFAULT NULL,

`expense\_date` date DEFAULT NULL,

`amount` decimal(10,2) DEFAULT NULL,

`description` text,

PRIMARY KEY (`user\_expense\_id`),

KEY `user\_id` (`user\_id`),

KEY `category\_id` (`category\_id`),

CONSTRAINT `user\_expenses\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`),

CONSTRAINT `user\_expenses\_ibfk\_2` FOREIGN KEY (`category\_id`) REFERENCES `budget\_categories` (`budget\_category\_id`)

)

`utility\_bills` (

`utility\_bill\_id` int unsigned NOT NULL AUTO\_INCREMENT,

`user\_id` int unsigned DEFAULT NULL,

`utility\_company` varchar(255) DEFAULT NULL,

`bill\_date` date DEFAULT NULL,

`amount` decimal(10,2) DEFAULT NULL,

`category` varchar(255) DEFAULT NULL,

PRIMARY KEY (`utility\_bill\_id`),

KEY `user\_id` (`user\_id`),

CONSTRAINT `utility\_bills\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`)

)

## 

## SDLC Process:

### Phase 1: Planning

* Objective: Define the project scope, requirements, and feasibility.
  + Activities:
    - Define Project Scope:
      * Collaborate with stakeholders to clearly outline the project boundaries.
      * Use Jira and Confluence for capturing and organizing project requirements.
    - Feasibility Study:
      * Conduct a detailed analysis of technical, operational, and economic feasibility.
      * Microsoft Project helps in creating a realistic project timeline and resource allocation.
  + Tools:
    - Jira: For project management and issue tracking.
    - Confluence: To collaborate on requirements and documentation.
    - Microsoft Project: For creating project timelines and Gantt charts.

### Phase 2: Analysis

* Objective: Gather and analyze requirements in detail.
  + Activities:
    - Requirements Gathering:
      * Interview stakeholders, conduct surveys, and utilize Google Forms for user feedback.
      * Document requirements with the help of Confluence and Jira.
    - User Feedback:
      * Design and distribute surveys using Google Forms or SurveyMonkey.
      * Incorporate user feedback into the requirement specifications.
  + Tools:
    - Figjam: For creating visual diagrams such as flowcharts and data models.
    - Figma: For creating interactive prototypes.
    - Google Forms: To gather user feedback and requirements.
    - Jira: Project management.
    - Confluence: Project documentation & knowledge base.

### Phase 3: Design

* Objective: Create detailed system design based on analyzed requirements.
  + Activities:
    - User Interface (UI) Design:
      * Create wireframes and interactive prototypes using Figma.
      * Share and iterate on designs with stakeholders.
    - System Architecture Design:
      * Utilize Enterprise Architect to create detailed system architecture and UML diagrams.
      * Share architecture documentation on Confluence for team collaboration.
  + Tools:
    - Figma: For designing the user interface and user experience.
    - Enterprise Architect: For creating detailed system architecture and UML diagrams.
    - Draw.io: For additional diagramming needs.

### Phase 4: Implementation (Coding)

* Activities:
  + Coding Standards:
    - Establish coding standards and guidelines for the team.
    - Use IDEs (Integrated Development Environments) like Visual Studio to enforce coding standards.
  + Version Control:
    - Implement branching strategies using Git for efficient collaboration.
    - Conduct regular code reviews through pull requests.
  + Build Automation:
    - Set up Jenkins or Travis CI for continuous integration to catch integration issues early.
* Objective: Translate design specifications into working code.
  + Tools:
    - IDEs (Integrated Development Environments):
      * Visual Studio
    - Version Control:
      * Gitlab: For source code version control.
    - Build Automation:
      * Gitlab CI: For continuous integration and automated builds.

### Phase 5: Testing

* Objective: Verify that the software meets the specified requirements.
  + Activities:
    - Unit Testing:
      * Develop and run unit tests using JUnit or TestNG.
      * Integrate automated tests into the build process.
    - Integration and System Testing:
      * Use Selenium for automated web application testing.
      * Postman for API testing and integration tests.
    - Test Case Management:
      * Organize and manage test cases in Jira (Zephyr).
      * Execute test cases and report issues.
  + Tools:
    - JUnit or TestNG: For unit testing (Java).
    - Selenium: For automated web application testing.
    - Postman: For API testing.
    - Zephyr: For test case management.

### Phase 6: Deployment

* Objective: Release the software to production.
  + Activities:
    - Containerization:
      * Containerize the application using Docker.
      * Define deployment configurations for different environments.
    - Continuous Deployment:
      * Implement GitLab CI/CD pipelines for automated deployment.
      * Utilize Kubernetes for container orchestration.
  + Tools:
    - Docker: For containerization.
    - Kubernetes: For container orchestration.
    - GitLab CI/CD pipelines: For continuous deployment.

### Phase 7: Maintenance and Support

* Objective: Provide ongoing support, address issues, and release updates.
  + Activities:
    - Monitoring and Performance:
      * Set up monitoring tools like or Datadog.
      * Regularly analyze performance metrics and address bottlenecks.
    - Issue Resolution:
      * Use Jira Service Management to manage and prioritize support tickets.
      * Regularly release updates with bug fixes and new features.
  + Tools:
    - Jira Service Management (formerly Jira Service Desk): For managing support tickets.
    - or Datadog: For monitoring and performance management.
    - Snyk: For continuous code quality inspection.