TASK 81HD

Software Project Planning, Design and Quality Management

SWE30010 - Managing IT Projects

Class: Fri 08:00 DT7.2 **- Tutor:** Pham Thi Kim Dung

Name: Trac Duc Anh Luong - **ID**: 103488117

Project description

Project Name: Speakeasy-OMS

Company: Speakeasy Cocktail Bar

Software: Order Management System (Speakeasy-OMS)

Synopsis
Speakeasy is a popular cocktail bar in Hanoi. We offer various cocktails and drinks with different variations from all over the world, with the choice to add different touches and customisations for our customers.

The bar currently uses physical queues and notes to take orders from customers. It is time-consuming and labour-costly for the bartenders and waiters to remember orders and manually enter the order information into Excel spreadsheets. It is also difficult for new customers who do not have a favourite drink in mind and need to ask the waiter or bartender for advice.

We would like a web app that lets customers look up drinks based on their preferences and make orders, as well as customised dashboards, order management and data analytics for the managers to monitor.

The web app must have a responsive user interface that is easy to use and a separate admin panel.

Background
Speakeasy is a well-established cocktail bar in Hanoi, known for its extensive menu of international drinks and customizability options. Currently, the bar relies on a manual system of queues, written notes, and Excel spreadsheets to manage orders. This process proves inefficient, requiring significant staff time to handle order taking, memorisation, and data entry. Furthermore, new customers often struggle to navigate the vast drink selection without staff assistance. To address these limitations, Speakeasy seeks a web application to streamline operations. This app will empower customers to browse drinks based on preferences, place orders directly, and personalise their selections. Additionally, Speakeasy desires a dedicated management interface for functionalities like order monitoring, data analysis, and customised dashboards. A user-friendly responsive interface and a separate admin panel are key priorities for this web application.

Backlog items list

No.	Item	Dependencies	Business Value (1 least – 10 most)	Release Schedule (Sprint 1 | 2 | 3 | …)
F1	Design database schema	None	9	Sprint 1
F2	Design UI wireframes and interactive prototypes for the app	None	8	Sprint 1
F3	Setup development environment	F1	7	Sprint 1

	and modules			
F4	Setup user roles (Customer & Admin), Registration and Login Screens	F3	7	Sprint 1
F5	Setup API connector in IS (Integration Service) module	F3	8	Sprint 1
F6	Create customer screens for searching and filtering drinks using API connector	F5	9	Sprint 1
F7	Add order mechanism and QR code for payment integration	F6	7	Sprint 2
F8	Create customer screens for favourite and drink history	F7	7	Sprint 2
F9	Create admin dashboard for analytics and Excel export	F8	8	Sprint 2
F10	Finish documentation: readme, basic design, detail design, etc.	F9	8	Sprint 2
F11	Deploy app into production	F10	10	Sprint 2
F12	Monitor app performance, gather feedback and reviews for future improvements	F11	9	Sprint 2

Figure 1: Backlog items and release schedules

Deliverables

1. Online cocktail order website (fully functional application)
2. Source code (all binaries, config files)
3. System documentation: readme, basic design, detail design, config instructions, etc.
4. Comprehensive admin manual (online & downloadable)
5. Detailed system training program for admin
6. Ongoing maintenance and support

Release schedule

The project will implement **2-week sprints** from April 1 to April 26. The details are as follows:

Sprint 1: April 1 – April 12
Sprint 2: April 15 - April 26

Software design

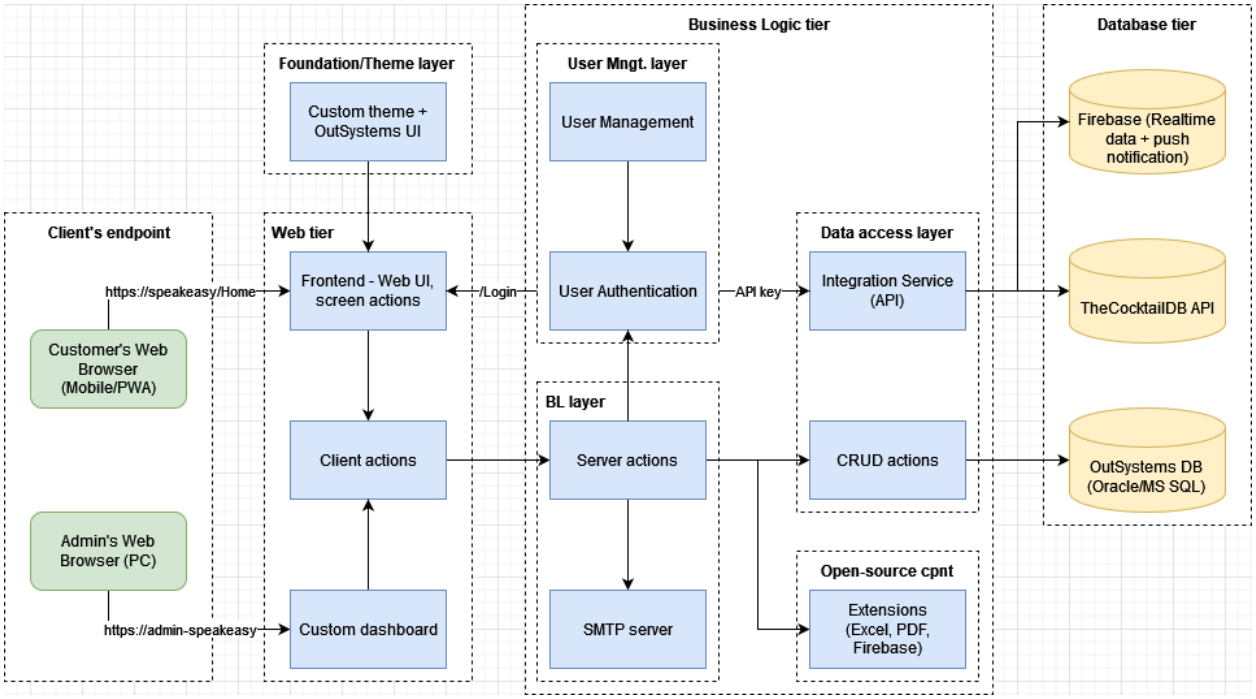

Figure 2: Software high-level architecture

In Figure 2, we can see the high-level implementation and interactions between different components of the Speakeasy-OMS software, following a 3-tier architecture.

1. Web tier: The UI components of the system
    a. Frontend - Web UI, screen actions: Include web pages, compiled HTML, CSS, JS, and screen actions that call to client and server actions. This is the customer's entry to the web page.
    b. Client actions: Perform system operations on the client side, call to server actions
    c. Custom dashboard: Entry to the admin's panel, including reports, statistics, and data export functions.
2. Business Logic tier: Server actions that validate data and communicate with the data access layer.
    a. User Management layer
        i. User Management: Provision of how users with different roles can access the screens following the least-privilege rule.
        ii. User Authentication: Validate OTPs and API keys sent from the client for login and accessing the Integration Service module.
    b. Business Logic layer
        i. Server actions: Contains server-side logic and business logic.
        ii. SMTP server: Send emails for OTP and customer service.
    c. Data Access layer
        i. Integration Service (API): Consumer for API to TheCocktailDB and Firebase extension in OutSystems.
        ii. CRUD actions: Performs basic CRUD actions on the OutSystems database.

d. Open-source components
    i. Extensions: Can implement from open-source Forge or self-custom extensions from the development team. Mostly used for PDF and Excel.

3. Database tier: Data storage for the system can be accessed from the Data Access layer in the BL tier.
    a. Firebase: Allows real-time data and push notifications.
    b. TheCocktailDB API: Public API that allows querying of data on drinks. The app will use a paid API key for upgraded features and performances.
    c. OutSystems DB: This database stores customer and order (invoice) information.

WBS and time estimation

For this section, we will provide the detailed plan and estimation for Sprint 1.

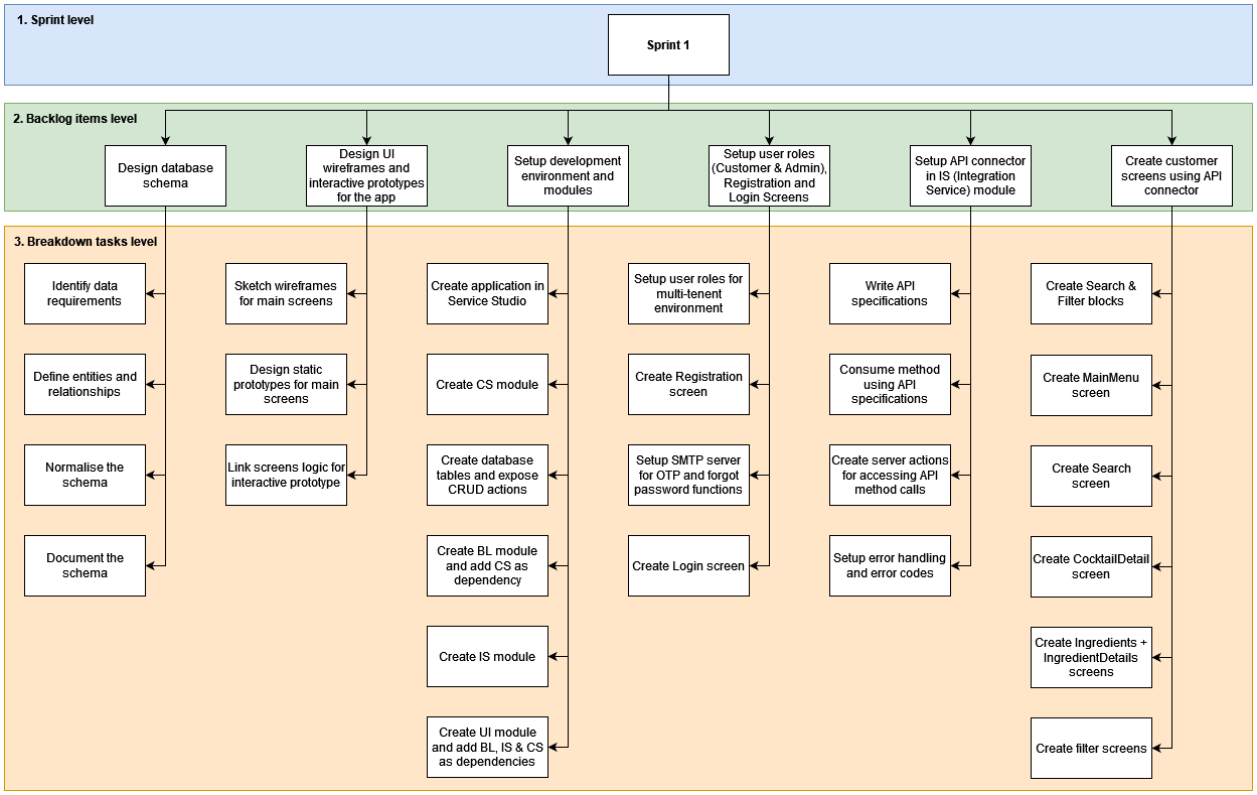

Figure 3: WBS tree structure

Notes:

1. Color annotation for Figure 3:
    a. Cyan: Sprint level
    b. Green: Backlog items level
    c. Orange: Breakdown tasks level
2. For planning, I will assume that the project development has 6 SWE30010 students, including my team members from Group 1 and me:
    a. Trac Duc Anh Luong (Self, PM)

b. Minh Nghia Nguyen (Scrum member)
c. Anh Duc Nguyen (Scrum member)
d. Gia Minh Nguyen (Scrum member)
e. Cong Anh Nguyen (Scrum member)
f. Tran Dat Dinh (Scrum member)

No.	Backlog item	Task	Assignee	Estimate
1	Design database schema	Identify data requirements	Trac Duc Anh Luong	2h
		Define entities and relationships	Minh Nghia Nguyen	2h
		Normalise the schema	Anh Duc Nguyen	2h
		Document the schema (Entity Design)	Gia Minh Nguyen	4h
2	Design UI wireframes and interactive prototypes for the app	Sketch wireframes for main screens	Cong Anh Nguyen	3h
		Design static prototypes for main screens	Tran Dat Dinh	6h
		Link screens logic for interactive prototype	Trac Duc Anh Luong	4h
3	Setup development environment and modules	Create the application in Service Studio	Minh Nghia Nguyen	1h
		Create CS module	Anh Duc Nguyen	1h
		Create database tables and expose CRUD actions	Gia Minh Nguyen	2h
		Create BL module and add CS as a dependency	Cong Anh Nguyen	1h
		Create IS module	Tran Dat Dinh	1h
		Create UI module and add BL, IS & CS as dependencies	Trac Duc Anh Luong	2h
4	Setup user roles (Customer & Admin), Registration and Login Screens	Setup user roles for multi-tenant environment	Minh Nghia Nguyen	2h
		Create Registration screen	Anh Duc Nguyen	6h
		Setup SMTP server for OTP and forgot password functions	Gia Minh Nguyen	3h
		Create Login screen	Cong Anh Nguyen	8h
5	Setup API	Write API specifications	Tran Dat Dinh	4h

	connector in IS (Integration Service) module	Consume method using API specifications	Trac Duc Anh Luong	4h
		Create server actions for accessing API method calls	Minh Nghia Nguyen	3h
		Setup error handling and error codes	Anh Duc Nguyen	4h
6	Create customer screens using API connector	Create Search & Filter blocks	Gia Minh Nguyen	5h
		Create MainMenu screen	Cong Anh Nguyen	3h
		Create CocktailDetail screen	Tran Dat Dinh	4h
		Create Ingredients & IngredientDetails screens	Trac Duc Anh Luong	6h
		Create Search screen	Minh Nghia Nguyen	8h
		Create filter screens	Anh Duc Nguyen	4h

Figure 4: WBS time/effort estimation

The task dependency is described in the table figure above. Some items can be done asynchronously; however, the logical sequence of tasks must be followed. The total estimated time for the project is 96 hours, matching the task requirement as follows:

- **6** SWE30010 students
- **8** hours of work per student
- **2**-week sprint

→ Calculation:
Total estimated time = [No. of students] x [hours of work per student] x [number of weeks per sprint]
= 6 x 8 x 2 = 96 (hours)

Definition of done

The Speakeasy-OMS project will follow the ISO/IEC 25010 quality standard to clearly define the DoDs.

No.	Characteristic	Sub-characteristic	Implementation
1	Functional sustainability	Functional completeness	Every backlog item and breakdown task in Sprint 1 is completed and tested by pre-defined test cases.
2		Functional correctness	No logical error after test cases are passed. Software degradation is closely monitored during the maintenance phase.
3		Functional appropriateness	Conducted usability tests must produce at least 4 out of 5 regarding user satisfaction.

4	Performance efficiency	Time behaviour	The average load time of the website should be under 2 seconds.
5		Resource utilisation	The server's RAM and CPU usage should not exceed 80% during heavy load.
6		Capacity	The system should be able to handle heavy traffic with 1000+ monthly users and handle concurrency effectively.
7	Compatibility	Co-existence	As OutSystems is a multi-tenant environment, effective access control is a must to prevent users from accessing unauthorised resources.
8		Interoperability	The admin panel will analyse all data generated by the system's order and customer's preferences.
9	Interaction capability	Appropriateness recognisability	Customer feedback on the experience of ordering in the bar should be more straightforward compared to when they had to wait in queues previously.
10		Learnability	The UI should have straightforward navigation and instructions made easy for first-time users.
11		Operability	The admin should be able to analyse the data with ease.
12		User error protection	All client and server actions must have error handling and rollback transactions if necessary.
13		User engagement	The UI and UX must be coherent and easy to use for the user.
14		Inclusivity	Customers can use the product from anywhere with localisation and drinks customised to their preferences.
15		User assistance	Guides and feedback hubs should be available for customers.
16		Self-descriptiveness	The usage and interactions of the web page should be obvious enough for any user.
17	Reliability	Faultlessness	99% of all edge cases covered, with 0 edge cases for payment transactions to be allowed.
18		Availability	The application has minimal downtime and is accessible to customers and admins most of the time.
19		Fault tolerance	The application can handle minor errors gracefully without complete system failure.
20		Recoverability	Implement data backup and recovery procedures to restore lost or corrupted information.
21	Security	Confidentiality	User data (account details, order history) is stored securely with appropriate access controls.

22		Integrity	Implement data validation techniques to prevent accidental or malicious data corruption.
23		Non-repudiation	Implement logging mechanisms to record user activity (e.g., order placement, admin actions).
24		Accountability	Logs record user activity with timestamps and identifying information.
25		Authenticity	Implement secure authentication mechanisms for user logins (e.g., strong passwords, two-factor authentication).
26		Resistance	Implement security measures to prevent common attacks like SQL injection, cross-site scripting (XSS).
27	Maintainability	Modularity	The application is designed with well-defined, independent modules with clear interfaces.
28		Reusability	Consider opportunities to create reusable components within the application (blocks).
29		Analysability	Use clear naming conventions, comments, and architecture diagrams for better code readability. AI Mentor should not report any architecture or convention issues.
30		Modifiability	The application's design and code should be adaptable to accommodate future changes and growth.
31		Testability	The application is designed to be easily testable at different levels (unit, integration, system).
32	Flexibility	Adaptability	Document any assumptions or dependencies regarding hardware, software, or operating systems.
33		Scalability	The application should be able to handle an increase in users and order volume without significant performance degradation.
34		Installability	As this is a web application, any operating system of device should be able to access it as long as an internet connection is established.
35		Replaceability	The focus of this project is likely to be building a new system, not replacing an existing one.
36	Safety	Operational constraint	Implement mechanisms to handle unexpected user inputs or errors gracefully.
37		Risk identification	Identify potential threats like unauthorised access, SQL injection, or data leaks.
38		Fail-safe	This involves features like automatic data backups, error logging, and restarting specific services in case of issues.

39		Hazard warning	The application should provide clear error messages and notifications to users in case of unexpected problems (no internet, new version updated, etc.).
40		Safe integration	Validate the security practices of any third-party components used.

Figure 5: DoD based on ISO/IEC 25010 characteristics

This DoD ensures the Speakeasy-OMS meets quality standards based on ISO25010 characteristics. Each backlog item should be considered "Done" only when it fulfils all the criteria mentioned above for its specific functionalities.