## \*\*System do obsługi zamówień baru FastFood\*\*

### \*\*Ogólna wizja\*\*

Celem projektu jest stworzenie systemu umożliwiającego szybkie i wygodne składanie zamówień w restauracji FastFood. System ma uprościć proces wyboru produktów, (obsługi płatności), odbioru zamówienia, jednocześnie integrując się z zapleczem kuchennym w celu efektywnego zarządzania kolejnością realizacji zamówień.

Aplikacja będzie składać się z trzech modułów:

1. \*\*Aplikacja mobilna/webowa dla klienta\*\* – składanie zamówień, (wybór metody płatności), śledzenie statusu.

2. \*\*Panel dla pracowników kuchni\*\* – wyświetlanie zamówień do realizacji w czasie rzeczywistym.

3. \*\*Panel administracyjny\*\* – zarządzanie menu, analizowanie statystyk zamówień.

---

## \*\*MVP (wersja na zaliczenie 3-4)\*\*

### \*\*Funkcjonalności podstawowe:\*\*

\*\*Moduł klienta:\*\*

- Przeglądanie menu (podstawowe produkty, brak konfiguracji dodatków).

- Składanie zamówienia (wybór produktów, podsumowanie, zatwierdzenie).

- Powiadomienie o statusie zamówienia (np. „w trakcie realizacji”, „gotowe do odbioru”).

\*\*Moduł kuchni:\*\*

- Panel wyświetlający zamówienia w kolejności zgłoszenia.

- Możliwość oznaczania zamówienia jako „w przygotowaniu” oraz „gotowe”.

\*\*Panel administracyjny:\*\*

- Podstawowe zarządzanie menu (dodawanie i usuwanie produktów).

### \*\*Aspekty inżynierii oprogramowania:\*\*

- Diagramy UML: przypadków użycia, klas, aktywności.

- Makieta interfejsu użytkownika (np. …).

- Wykorzystanie repozytorium kodu (Git) oraz systemu do zarządzania pracą (Jira).

- Testy jednostkowe (np. …) dla krytycznych funkcji backendu.

- Podstawowe testy funkcjonalne backendu (np. …).

---

## \*\*Rozszerzenia na wyższą ocenę (4.5-5.0)\*\*

### \*\*Dodatkowe funkcjonalności:\*\*

⭐ \*\*Dla klienta:\*\*

- Personalizacja zamówień (dodawanie/usuwanie składników).

- Historia zamówień i możliwość ich ponownego składania.

- Integracja z płatnościami online (np. BLIK, karta płatnicza).

⭐ \*\*Dla kuchni:\*\*

- System priorytetyzacji zamówień (np. osobno dla McDrive i zamówień na miejscu).

- Wskaźnik średniego czasu realizacji zamówień.

⭐ \*\*Dla administracji:\*\*

- Zaawansowana analiza sprzedaży (np. najczęściej wybierane produkty, godziny szczytu).

- Zarządzanie promocjami i rabatami.

### \*\*Zaawansowane aspekty inżynierii oprogramowania:\*\*

- Diagramy sekwencji i stanów.

- Testy integracyjne systemu (np. …).

- Pokrycie kodu testami (np. …).

- Optymalizacja wydajności bazy danych (np. indeksowanie)

**System do obsługi zamówień baru FastFood**

IO-09-ffOMS

**Kierownik:**

Szymon Łabędziewski

**Analityk:**

Wojciech Marchewka

Paweł Dyjak

Karol Starzyk

**Programista:**

Kacper Plutowski

Aleksander Kwaśny

Karol Starzyk

**Tester:**

Kacper Plutowski

Karol Starzyk

Aleksander Kwaśny

Wojciech Marchewka

Paweł Dyjak

**Inne:**

Szymon Łabędziewski

Kacper Plutowski

Karol Starzyk

Wojciech Marchewka

## **1. Sprint Backlog – Zadania zaplanowane na sprint**

**Zadanie 1: Konfiguracja repozytorium kodu (Git)**

* **Opis:** Utworzenie repozytorium, ustalenie struktury katalogów (foldery: klient, kuchnia, admin), wstępny plik README oraz konwencje commitów.
* **Odpowiedzialni:**
  + Kacper Plutowski
  + Aleksander Kwaśny
  + Karol Starzyk

**Zadanie 2: Konfiguracja Jira oraz stworzenie backlogu**

* **Opis:** Utworzenie projektu w Jira, zdefiniowanie epików, historyjek użytkownika oraz kryteriów akceptacji, przygotowanie sprint backlogu wraz z widocznym wykresem spalania.
* **Odpowiedzialni:**
  + Szymon Łabędziewski
  + Wojciech Marchewka
  + Paweł Dyjak

**Zadanie 3: Opracowanie MVP – Definicja funkcjonalności**

* **Opis:** Zdefiniowanie głównych funkcjonalności MVP obejmujących moduły klienta (przeglądanie menu, składanie zamówienia, powiadomienia), kuchni (lista zamówień, zmiana statusu) oraz administracji (zarządzanie menu).
* **Odpowiedzialni:**
  + Szymon Łabędziewski (opieka)
  + Wojciech Marchewka
  + Paweł Dyjak
  + Karol Starzyk

**Zadanie 4: Implementacja wersji bazowych modułów**

* **a) Moduł klienta:**
  + **Opis:** Implementacja interfejsu umożliwiającego przeglądanie menu, składanie zamówień oraz otrzymywanie powiadomień o statusie.
  + **Odpowiedzialni:**
    - Kacper Plutowski
    - Aleksander Kwaśny
  + **Planowany rezultat:** Działający prototyp umożliwiający składanie zamówień.
* **b) Moduł kuchni:**
  + **Opis:** Implementacja panelu dla pracowników kuchni do wyświetlania zamówień oraz zmiany ich statusu („w przygotowaniu”, „gotowe”).
  + **Odpowiedzialni:**
    - Karol Starzyk
    - Kacper Plutowski
  + **Planowany rezultat:** Podstawowy panel dla kuchni gotowy do pierwszych testów.
* **c) Panel administracyjny:**
  + **Opis:** Implementacja funkcji zarządzania menu – dodawanie, edycja, usuwanie produktów.
  + **Odpowiedzialni:**
    - Aleksander Kwaśny
    - Karol Starzyk
  + **Planowany rezultat:** Wstępna wersja panelu administracyjnego.

**Zadanie 5: Wstępna makieta (low-fi) interfejsu użytkownika**

* **Opis:** Przygotowanie szkiców dla wszystkich modułów (klient, kuchnia, admin) pokazujących układ przycisków i flow użytkownika.
* **Odpowiedzialni:**
  + Szymon Łabędziewski (opieka)
  + Kacper Plutowski
  + Karol Starzyk
  + Wojciech Marchewka
* **Planowany rezultat:** Interaktywny prototyp low-fi wstępnie zatwierdzony przez zespół.

**Zadanie 6: Diagram UML – Przypadki użycia**

* **Opis:** Opracowanie diagramu UML obejmującego główne przypadki użycia systemu (składanie zamówienia, zmiana statusu, zarządzanie menu).
* **Odpowiedzialni:**
  + Wojciech Marchewka
  + Paweł Dyjak
  + Karol Starzyk
* **Planowany rezultat:** Diagram UML przedstawiający scenariusze interakcji użytkowników z systemem.

**Zadanie 7: Flow użytkownika**

* **Opis:** Wizualizacja kroków od wejścia do aplikacji, wyboru produktów, składania zamówienia oraz śledzenia jego statusu.
* **Odpowiedzialni:**
  + Szymon Łabędziewski (opieka)
  + Karol Starzyk
  + Wojciech Marchewka
* **Planowany rezultat:** Graficzny flowchart z opisem poszczególnych kroków użytkownika.

## **2. Product Increment – Wykonane zadania**

### **> Konfiguracja repozytorium (Git)**

* **Odpowiedzialni:** Kacper Plutowski, Aleksander Kwaśny, Karol Starzyk
* **Status:** [Jira]
* **Link:** GitHub – Repo

### **> Konfiguracja Jira oraz stworzenie backlogu**

* **Odpowiedzialni:** Szymon Łabędziewski, Wojciech Marchewka, Paweł Dyjak
* **Status:** [Jira]

### **> Opracowanie MVP – Definicja funkcjonalności**

* **Odpowiedzialni:** Szymon Łabędziewski, Wojciech Marchewka, Paweł Dyjak, Karol Starzyk
* **Status:** [Jira]
* **Link:** Dokument MVP

### **> Implementacja wersji bazowej modułu klienta**

* **Odpowiedzialni:** Kacper Plutowski, Aleksander Kwaśny
* **Status:** [Jira]
* **Efekt:** Wstępnie działający interfejs do składania zamówień.
* **Link:** Commit – Moduł Klienta

### **> Implementacja wersji bazowej modułu kuchni**

* **Odpowiedzialni:** Karol Starzyk, Kacper Plutowski
* **Status:** [Jira]
* **Efekt:** Podstawowy panel kuchni umożliwiający wyświetlanie zamówień i zmianę statusu.
* **Link:** Commit – Moduł Kuchni

### **> Wstępna makieta interfejsu użytkownika (low-fi)**

* **Odpowiedzialni:** Szymon Łabędziewski, Kacper Plutowski, Karol Starzyk, Wojciech Marchewka
* **Status:** [Jira]
* **Efekt:** Prototyp interfejsu (low-fi) dostępny do przeglądu w Figmie.
* **Link:** Prototyp Low-Fi

### **> Diagram UML – przypadki użycia**

* **Odpowiedzialni:** Wojciech Marchewka, Paweł Dyjak, Karol Starzyk
* **Status:** [Jira]
* **Efekt:** Wstępny diagram UML do konsultacji z zespołem.
* **Link:** Diagram UML

### **> Flow użytkownika**

* **Odpowiedzialni:** Szymon Łabędziewski, Karol Starzyk, Wojciech Marchewka
* **Status:** [Jira]
* **Efekt:** Flowchart procesu użytkownika gotowy do dalszych poprawek.
* **Link:** Flowchart

## **3. Plan kolejnego sprintu (4 tygodnie)**

**Zadania do wykonania:**

1. **Opracowanie architektury systemu**
   * **Opis:** Zaprojektowanie modularnej architektury (opis warstw, komunikacja między komponentami).
   * **Odpowiedzialni:**
     + *Programiści*
   * **Efekt:** Dokument z architekturą systemu wraz z diagramem komponentów.
2. **Sfinalizowanie kryteriów akceptacji i scenariuszy testowych**
   * **Opis:** Uzupełnienie dokumentacji o kompletne kryteria akceptacji dla każdej funkcjonalności oraz opracowanie scenariuszy testowych (przynajmniej jeden dla każdego modułu MVP).
   * **Odpowiedzialni:**
     + Aleksander Kwaśny, Paweł Dyjak, Karol Starzyk
   * **Efekt:** Dokumentacja wymagań oraz zestaw przypadków testowych do weryfikacji funkcjonalności.
3. **Udoskonalenie Flow użytkownika i interaktywny prototyp (low-fidelity)**
   * **Opis:** Rozbudowa i poprawa istniejącego flow użytkownika oraz aktualizacja prototypu interfejsu w narzędziu z dodatkowymi informacjami od testerów i użytkowników.
   * **Odpowiedzialni:**
     + Szymon Łabędziewski (UX)
     + Wojciech Marchewka
   * **Efekt:** Gotowy interaktywny prototyp low-fi wspierający dalsze iteracje UI.
4. **Prezentacja wykresu spalania sprintu w Jira**
   * **Opis:** Zaktualizowanie i udostępnienie wykresu burndown w Jira z podsumowaniem postępów sprintu.
   * **Odpowiedzialni:**
     + Szymon Łabędziewski
   * **Efekt:** Widoczny, aktualny wykres spalania, który umożliwia ocenę tempa realizacji zadań.

## **4. Artefakty (obecne i wymagane)**

* **Jira:**
  + Skonfigurowana tablica sprintu z backlogiem, historyjkami użytkownika, kryteriami akceptacji oraz wykresem spalania.
* **Repozytorium kodu (Git):**
  + Kod z podziałem na moduły, uporządkowany README oraz konwencjonalne komunikaty commit.
* **Prototyp interfejsu (low-fi):**
  + Interaktywny prototyp w pokazujący podstawowe widoki dla użytkownika, kuchni i panelu administracyjnego.
* **Diagram UML:**
  + Wstępny diagram przypadków użycia (ew. diagram klas/aktywności).

## **Rozpiska zadań i ustaleń dla kolejnej fazy projektu**

### **1. Artefakty do dostarczenia**

**1.1 Architektura systemu**

• Diagram komponentów (frontend, backend, baza danych, kolejki, integracje)

• Diagram pakietów / modułów

• Opis warstw (prezentacji, logiki biznesowej, dostępu do danych)

• Wybór technologii i uzasadnienie (React Native vs. web, Node.js vs. Java, PostgreSQL vs. NoSQL)

= Karol Starzyk, Aleksander Kwaśny

**1.2 Kompletny dokument wymagań + scenariusze testowe**

• Zestaw user stories z pełnymi kryteriami akceptacji

• Specyfikacja funkcjonalna i niefunkcjonalna (wydajność, bezpieczeństwo, skalowalność)

• Scenariusze testowe (jednostkowe, integracyjne, akceptacyjne) dla każdej historii

= Wojciech Marchewka, Paweł Dyjak

**1.3 Interaktywny prototyp IO (low-fi ➔ mid-fi)**

• Kluczowe ekrany klienta, kuchni i admina

• Nawigacja i przepływ (klikalne hotspoty)

• Uwzględnienie wyników badań UX (problemy, sugestie)

= Szymon Łabędziewski, Kacper Plutowski

### **2. Zadania do realizacji**

#### **2.1 Badania interfejsu w gronie użytkowników**

1. **Rekrutacja**: dobór 5–7 użytkowników (klienci, obsługa)
2. **Scenariusze testowe**: zadania – złożenie zamówienia, sprawdzenie statusu
3. **Sesje**: notatki, obserwacje punktów krytycznych
4. **Raport z badań**: wnioski i rekomendacje do prototypu

*Odpowiedzialni:* Aleksander Kwaśny, Karol Starzyk

#### **2.2 Diagramy projektowe**

* **Diagram klas**: definiuje encje domenowe (Produkt, Zamówienie, Użytkownik, Sesja)
* **Diagram pakietów**: podział kodu na moduły (customer, kitchen, admin, common)
* **Diagram bazy danych**: tabele, relacje, klucze obce
* **Diagramy sekwencji** (opcjonalnie):
  + Składanie zamówienia (frontend → backend → DB → queue → kuchnia)
  + Aktualizacja statusu (kuchnia → backend → frontend)

*Odpowiedzialni:* Wojciech Marchewka, Paweł Dyjak

#### **2.3 Implementacja aplikacji**

1. **Frontend**
   * Struktura projektu, routing, komponenty UI
   * Integracja z API (REST / GraphQL)
2. **Backend**
   * Definicja modeli danych, kontrolery, serwisy
   * Autoryzacja i uwierzytelnianie
3. **Baza danych**
   * Schemat, migracje, optymalizacje indeksów

*Odpowiedzialni:* – Frontend: Aleksander Kwaśny, Wojciech Marchewka  
 – Backend: Karol Starzyk, Kacper Plutowski

#### **2.4 Testy**

* **Testy modułowe**: każdy serwis / funkcja backendu
* **Testy integracyjne**: przepływ całego zamówienia między modułami
* **Testy funkcjonalne (E2E)**: symulacja scenariuszy z wykrytych błędów UX

*Odpowiedzialni:* – Testerzy: Paweł Dyjak, (Aleksander Kwaśny, Wojciech Marchewka)

### **3. Harmonogram i priorytety (kolejność działań)**

1. **Tydzień 1**
   * Rekrutacja i przygotowanie badań UX
   * Szkic diagramów pakietów i bazowy komponent diagramu klas
2. **Tydzień 2**
   * Sesje UX i raport
   * Finalizacja diagramów klas i bazy danych
   * Start implementacji podstawowych modeli i usług backendu
3. **Tydzień 3**
   * Budowa prototypu mid-fi w Figmie z naniesionymi uwagami
   * Implementacja endpointów REST i pierwsze testy modułowe
4. **Tydzień 4**
   * Ukończenie interaktywnego prototypu
   * Testy integracyjne i funkcjonalne
   * Podsumowanie: weryfikacja architektury, scenariuszy, artefaktów, przygotowanie do demo

## **1. Sprint Backlog (zadania)**

1. **Konfiguracja repozytorium i infrastruktury**
2. **Implementacja modułu administracyjnego** (admin.html + połączenie z Supabase)
3. **Implementacja modułu klienta** (client.html + składanie zamówień)
4. **Implementacja modułu kuchni** (kitchen.html + obsługa statusów)
5. **Centralny interfejs** (index.html z zakładkami i iframe)
6. **Schemat bazy danych** (db.sql)
7. **Diagramy UML** (klas, aktywności, przypadków użycia)
8. **Kryteria akceptacji i scenariusze testowe**

## **3. Plan na następne 4 tygodnie**

1. **Architektura systemu**
   * Diagram komponentów i pakietów
   * Opis warstw i technologii
2. **Pełna specyfikacja wymagań**
   * Rozszerzenie user stories o niefunkcjonalne
   * Uzupełnienie kryteriów akceptacji
3. **Scenariusze testowe**
   * Testy integracyjne
4. **Interaktywny prototyp (mid-fi)**
   * Rozbudowa low-fi w Figmie aby łapał wszystkie ścieżki użytkownika
   * Uwzględnienie wyników badań UX
5. **Implementacja kolejnych funkcji**
   * Priorytetyzacja zamówień, historia zamówień
6. **Testy i optymalizacje**
   * Testy
   * Optymalizacja zapytań do bazy i indeksów