[미스터 대박 디너 서비스] 프로젝트 계획서

Team 뚝배기
최문기, 유시온

1. 개요

1.1. 프로젝트 개요

본 프로젝트의 목적은 '미스터 대박' 비즈니스를 지원하기 위한 시스템을 만드는 것이다. '미스터 대박' 비즈니스는 훌륭한 식사를 배달해주는 서비스로 '발렌타인 디너', '프렌치 디너', '잉글리시 디너', '샴페인 축제 디너' 등의 메뉴를 주문하면 각 메뉴에 맞는 음식과 각종 소품이 배달되는 방식이다. 각 메뉴 주문 시 스타일을 'simple', 'grand', 'deluxe'로 지정할 수 있고, 각 메뉴 별로 기본 제공되는 음식의 양을 변경하거나 특정 음식을 삭제할 수 있다.

본 프로젝트에서는 '미스터 대박' 비즈니스에 사용할 주문 시스템을 제작해야 한다. 고객을 위한 인터페이스와 직원을 위한 인터페이스로 총 두 개의 인터페이스가 필요하다.

고객 인터페이스에서는 메뉴와 스타일을 선택하고 음식을 변경할 수 있는 주문 기능, 최근 주문 내역을 조회할 수 있는 기능, 단골 고객에게 할인을 제공하는 기능 등을 제공할 것이다.

직원 인터페이스에서는 고객의 주문을 확인하고 "조리 전", "조리 중", "배달 중"으로 주문의 상태를 전환하여 관리하는 기능을 제공할 것이다. 그리고 완료된 주문 내역을 확인하는 기능을 제공할 것이다.

1.2. 프로젝트의 산출물

2020년 11월 6일까지 중간 보고서와 UML 다이어그램을 산출물로 제출할 예정이다. 최종적으로 12월 3일까지 구현을 마치고 데모 프로그램과 최종보고서를 제출할 것이다.

일정	산출물
프로젝트 계획	프로젝트 계획서
중간 보고서 (11월 6일)	분석산출물 - 유스케이스도 - 유스케이스 설명서 - 액티비티도 설계산출물 - 아키텍처도 - 서브시스템 세부 설계 - 교류도 - 상태도

최종 보고서 (12월 3일)	구현 및 시험산출물 - 컴포넌트도 - 시험 보고서 - 오류 보고서 운영산출물 - 배치도

1.3. 정의, 약어

본 계획서에서 '서비스'라 함은 본 프로젝트의 결과물로 나오는 시스템을 뜻한다.

2. 자원 및 일정 예측

2.1. 자원

가. 인력

총 두 명의 인력이 투입될 예정이다.

나. 비용

PERT(Program Evaluation and Review Technique)로 계산한 소요 기간의 기대치는 다음과 같다.

(a) 낙관적(Optimistic) 소요 기간 : 40일

(m) 보통(Most likely) 소요 기간 : 49일

(b) 비관적(Perssimistic) 소요 기간 : 52일

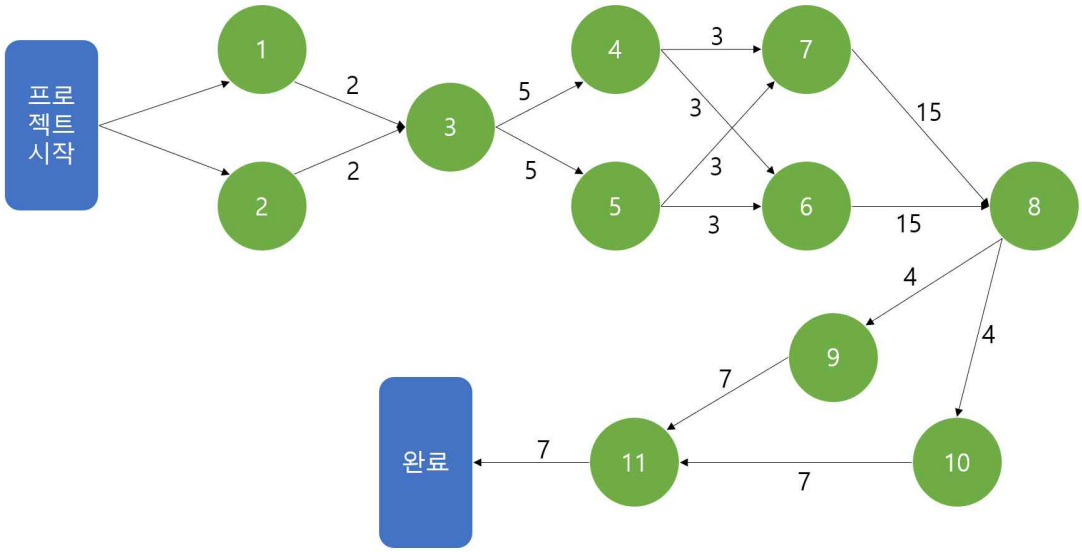
t_e = (a + 4m + b)/6 = 48일

2.2. 일정

작업에 따른 선행 작업과 소요 기간은 다음과 같다.

작업	선행 작업	소요 기간(일)
1. 시나리오 분석	-	2일
2. 성능 요구 분석	-	2일
3. 시스템 구조 설계	1, 2	5일
4. 모듈 설계	3	3일
5. DB 설계	3	3일
6. 프론트엔드 개발	4, 5	15일
7. 백엔드 개발	4, 5	15일
8. 통합	6, 7	4일
9. 시스템 테스트	8	7일
10. 인수 테스트	8	7일
11. 모듈 테스트	8	7일

CPM 네트워크로 그려보면 아래와 같고 임계경로의 소요시간은 43일이다.



▲ CPM 네트워크

전체 서비스 개발 일정은 총 7주로 계획하였다. 간트 차트로 그려보면 다음과 같다.

▲ 전체 서비스 개발 일정

3. 팀 구성 및 인력 배치

3.1. 팀 구성

팀명은 뚝배기이며 팀장 컴퓨터과학부 최문기, 팀원 컴퓨터과학부 유시온으로 구성되어 있다. 인력 배치는 다음과 같다.

직능	팀원
프로젝트 관리자	최문기
시스템 설계자	유시온
웹 개발자 및 디자이너	최문기
데이터베이스 및 시스템 개발자	유시온
QA 관리자	최문기, 유시온

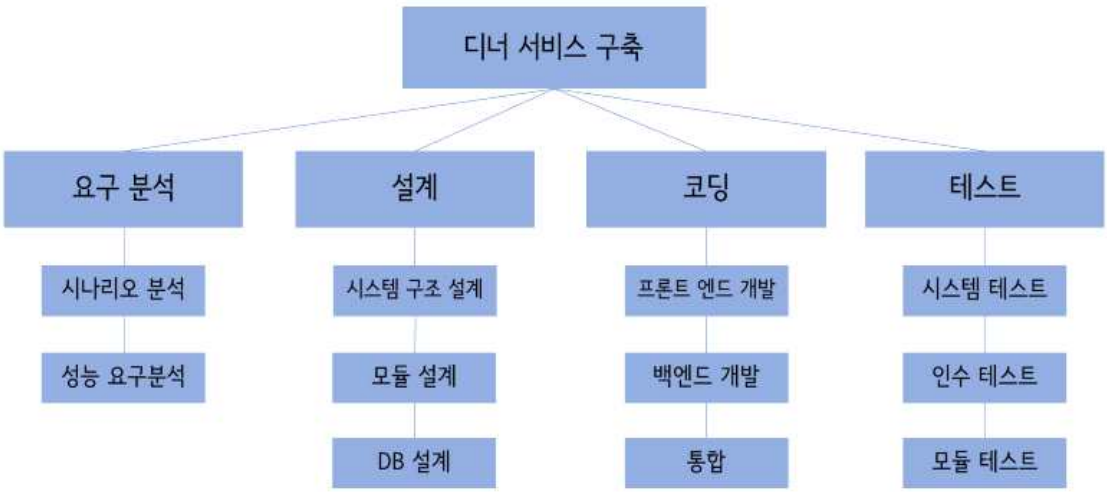
3.2. 직무 기술

최문기 : html, css, javascript, bootstrap

유시온 : node.js, express, javascript

4. WBS(Work Breakdown Structure)

WBS(Work Breakdown Structure)를 계층화하여 표현하면 다음과 같다.



5. 기술 관리 방법

5.1. 변경 관리

Git을 이용하여 master 브랜치는 항상 릴리스할 수 있는 버전으로 유지하고 각 기능을 개발할 때는 새로운 브랜치에 커밋하여 기능 개발 후 테스트까지 완료되고 나서 최종적으로 master 브랜치에 합병할 것이다.

또한, 커밋메시지 규칙을 정하여 팀원 간 소통 및 버전 관리에 어려움이 없게 할 예정이다.

5.2. 위험 관리

프로젝트 진행 과정에서 문제가 발생하면 프로젝트 계획에 차질이 생기지 않도록 바로 파악하고 처리해야한다. 문제 발생 시에 프로젝트 진행 인원 사이에서 신속한 의사소통을 통해서 문제를 고칠 수 있도록 하며 미리 위험 요소를 분석하고 해결 방안을 마련한다.

5.3. 비용 및 진도 관리

본 프로젝트는 매 주 달성하고자 하는 목표를 설정하여 계획대로 잘 되고 있는지 주 단위로 체크할 것이며 진행도에 따라 각자의 작업량을 늘릴 수도 있고 줄일 수도 있다.

5.4. 위험 분석 및 해결 방안

현재 파악되는 위험으로는 중간고사, 기말고사 시험 대비, 다른 과목의 팀 프로젝트, 과제, 잘못된 기능, 인터페이스 개발, 설계 오류, 모델링 오류, 기술 오류가 있다.

위험요소	평가	해결방안
중간고사, 기말고사, 타 과목 프로젝트, 과제	high	시험과 같이 예측할 수 있는 일정에 대해서는 미리 고려하여 관리한다. 초기 계획을 여유롭게 설정한다.
잘못된 기능, 인터페이스 개발	Extremely high	요구사항을 분석할 때 확실하게 검토하여 분석하고 변경이 있을 경우를 미리 대비한다.
설계 오류, 모델링 오류	Extremely high	초기에 발생한 설계 오류와 모델링 오류는 늦게 발견할 경우 전체 진행 과정에 끼치는 영향이 크기 때문에 최대한 확실하게 설계하고 논의한다.
기술 오류	high	백엔드에서는 Node.js로 개발할 때에는 console.log()로 메시지를 띄워주어 디버깅을 하기 쉽도록 한다. 프론트 엔드에서는 테스팅 코드를 작성한다. javascript로 개발하는데 undefined나 NaN 타입이 있음에도 불구하고 동작할 수도 있지만 undefined나 NaN 타입은 절대로 발생하지 않도록 하여 추후에 있을 에러에 대비한다.

▲ 위험 분석 및 해결 방안

6. 표준 및 개발 절차

6.1. 코딩 표준

코드 스타일은 Google JavaScript Style Guide[1]에 따른다.

6.2 Rules by identifier type

6.2.1 Package names

Package names are all `lowerCamelCase`. For example, `my.exampleCode.deepSpace`, but not `my.examplecode.deepspace` or `my.example_code.deep_space`.

6.2.2 Class names

Class, interface, record, and typedef names are written in `UpperCamelCase`. Unexported classes are simply locals: they are not marked `@private` and therefore are not named with a trailing underscore.

Type names are typically nouns or noun phrases. For example, `Request`, `ImmutableList`, or `VisibilityMode`. Additionally, interface names may sometimes be adjectives or adjective phrases instead (for example, `Readable`).

6.2.3 Method names

Method names are written in `lowerCamelCase`. Names for `@private` methods must end with a trailing underscore.

Method names are typically verbs or verb phrases. For example, `sendMessage` or `stop_`. Getter and setter methods for properties are never required, but if they are used they should be named `getFoo` (or optionally `isFoo` or `hasFoo` for booleans), or `setFoo(value)` for setters.

Underscores may also appear in JsUnit test method names to separate logical components of the name. One typical pattern is `test<MethodUnderTest>_<state>_<expectedOutcome>`, for example `testPop_emptyStack_throws`. There is no One Correct Way to name test methods.

▲ Google JavaScript Style Guide

6.2. 개발 방법론 및 절차

객체 지향 방법론을 사용한다. UML(Unified Modeling Language)를 사용하여 소프트웨어를 모델링하며 starUML을 사용한다. 개발 프로세스는 폭포수 모델을 따라서 이전 프로세스를 완전히 완료하고 다음 프로세스를 진행한다. 각각의 프로세스가 완료될 때는 다음 프로세스의 진행에 문제가 발생하지 않도록 철저하게 수행하며 검토한다.

6.3. 사용 라이브러리, 프레임워크

node.js, express, javascript, bootstrap

7. 검토 회의

7.1. 검토회 일정

매 기능을 구현할 때마다 먼저 구현자가 테스트를 수행해야 한다. 구현자의 테스트가 끝나면 제3자가 테스트를 한다. 이 테스트 결과를 토대로 기능에 대해 검토회를 할 것이다. 검토회는 매주 월요일, 수요일, 금요일 오후 9시에 할 예정이다.

7.2. 검토회 진행 방법

검토회는 온라인으로 진행할 것이며 늘어짐을 방지하기 위해 당일의 목적을 세우고 절차에 맞춰 진행할 것이다. 온라인 화상 회의 플랫폼으로는 구글미트를 사용한다.

---

1) https://google.github.io/styleguide/jsguide.html

7.3. 검토회 후속 조치

검토회가 끝나고 나면 팀원 유시온이 검토회의 내용을 정리하여 수정 사항과 추가 사항에 대하여 다음 기능 구현 시에 덧붙일 수 있도록 한다.

8. 개발 환경

프레임워크 및 라이브러리	사용 프로그램
html, css, javascript, bootstrap	Visual studio code
node.js, express	Visual studio code

9. 성능 시험 방법

먼저 테스트 케이스를 생성해서 시스템 상으로 어떠한 오류가 없는지 확인한다. 블랙박스 테스트, 동등 분할 기법, 경곗값 분석 등의 방법을 통해서 확인할 예정이다. UI에 어떠한 결함이 없는지 확인한다. UI 테스트는 자동화할 수 없기에 팀원 두명이 직접 확인한다. 최종적으로는 인수테스트를 수행하며 알파 테스트와 베타 테스트로 나눠서 수행한다.

10. 문서화

분석산출물, 설계산출물, 구현 및 시험산출물, 운영산출물과 API 명세 등을 문서화한다. 각종 산출물은 word로 작성하고 API 명세는 excel을 이용한다. 그리고 회의록과 같은 모든 기타 문서는 markdown으로 작성하며 GitHub Flavored Markdown Spec[2]에 따라 작성한다. 불필요한 반복을 피해 간결하게 작성하며 읽기 쉽도록 작성한다.

11. 유지 보수

유지보수 프로세스로는 반복적 개선 모델을 도입할 예정이다. 서비스에 문제가 발견될 시, 분석, 재설계, 구현의 단계를 거쳐 프로그램을 점점 개선해 나갈 것이다. 또한 처음부터 시스템 구축 시에 모듈화를 철저히 하여 유지보수에 드는 비용을 줄일 것이다.

문제를 수정하고 통합하는 과정은 Git을 이용하여 새로운 브랜치를 만들고, master 브랜치에 병합하는 과정을 통하여 실현할 것이다.

---

2) https://github.github.com/gfm/

12. 설치, 인수

개발이 완료되면 사용자에게 인수하는 과정을 거친다. 이 때 인수 테스트 수행할 것이며 알파 테스트와 베타 테스트를 통해 외부 환경에서 서비스가 잘 작동하는지 확인하고 테스터로부터 문제를 보고받을 것이다. 발생한 문제에 대한 변경 과정을 거치고 최종적으로 서비스를 인수할 예정이다.

13. 참고문헌 및 부록

[1] Google JavaScript Style Guide https://google.github.io/styleguide/jsguide.html
[2] GitHub Flavored Markdown Spec https://github.github.com/gfm/