BTE 533 – Software Engineering

Taxi Booking Application (Uber)

Section 1 Project Plan Document

Student Name: Harun Murat Bulut
Student Number: 0701201015
Submission Date: 15.06.2021

1. INTRODUCTION

Project plan document of Taxi Booking Application provides an overview of the purpose, scope and objectives of the project. This document includes project plan, a list of project deliverables, project schedule, estimation, team structure and project risk.

1.1 Scope

In this project, there would be new mobile application and with that application any driver can be able to use that and make taxi booking with their personal car. Any passenger should be able to use this application to travel any point to another point. GPS would be used for finding optimum match of user and driver. There should be also price calculator in this application. Overall Project documents are listed below:

1. Project Plan
2. Requirements Specification Document
3. Analysis Document
4. Design/Architecture Document
5. Test Document

1.2 Deliverables

1. Preliminary Project Plan
2. Requirements Specification
3. Analysis
4. Architecture Specification
5. Source Code
6. Test Plan
7. Final Product w/ Demo
8. Documentation

1.3 Epics

- Booking a car
- Finding passenger
- Rating drivers
- Tracking car
- Make a payment
- Registration for drivers and users
- Messaging between drivers and users

1.4 Non-functional issues

Usability	Friendly GUI's
Accessibility	Security
Performance	Maintainability
Speed	Extensibility and Stability
Efficiency	Disaster Recovery
Availability	Error Handling

1.4.1 Usability
The Application should be easy to use by every user. In order to accomplish this objective, the system should have a simple and well-designed interface.

1.4.2 Accessibility
The Application should be made accessible to the people who live everywhere in the world. anyone can use the system regardless of the location and can get the information they acquire.

1.4.3 Performance
The performance of the application should be fast and efficient in adding information of drivers. The system should be available for user in real time and always up to date.

1.4.4 Speed
The application response time is a significant requirement because the action cannot be postponed or delayed. The application should be fast enough to satisfy the user's needs and should not waste their time.

1.4.5 Efficiency
Efficiency of any system is concerned with the minimum processing time as well as the optimal use of system resources in designing the proposed systems. Our android application will be efficient in using processing resources. It can be efficiently run on all android devices.

1.4.6 Availability
The Application should operate 24 hours a day.

1.4.7 Friendly GUI's
The users of this application have different types of people and different levels of technical skills, therefore the application should be understandable by all the users. Consequently, the Application should provide an easy to use, friendly Graphical User Interface (GUI).

1.4.8 Security
Data inserted by user is secured and saved by this application, and will be redundant, in order to perform the exact action in specified situation. The payment security is also so important and must have been take consideration.

1.4.9 Maintainability
The application should be maintained in order to perform the best of its ability.

1.4.10 Extensibility and Stability
The application should be flexible enough to allow improvements for the future and should be able to adapt any additional future change in activities; the application components can be modified for more changes and features allow the addition of new features without disturbing the main functionalities of the application.

1.4.11 Disaster Recovery
The application should be able to recover from an unsuitable problem .and should back up data.

1.3.12 Error Handling
The application should be able to handle unexpected errors quickly and easily.

2. PROJECT PLAN

1. The customer can request a car via the app, either instantly or scheduled. Drivers should answer requests. **(XL)**

2. The driver has the option to accept or reject the booking. If a driver refuses a booking, then the next available driver may approve the request. **(S)**

3. The passenger should track the car and check out the estimated time of arrival. **(M)**

4. The app helps to calculate the price beforehand, and the customer can pay with their credit card. **(M)**

5. Passengers can make comment to driver and give them a rating as well. **(S)**

6. Messaging should be done between passenger and driver. **(S)**

7. There would be a separate UI and logic for both drivers and passenger, there would be 2 registration option: Driver and Passenger. **(M)**

8. Driver should monitor nearest passengers and make a booking offer to them. **(M)**

3. ESTIMATES

Calculation of Function Point (FP)

Function Point (FP) is an element of software development which helps to approximate the cost of development early in the process. It may measure functionality from user's point of view. With FP approach we can determine the size of project in terms of Line of Effective Code (LOC).

Counting Function Point (FP):

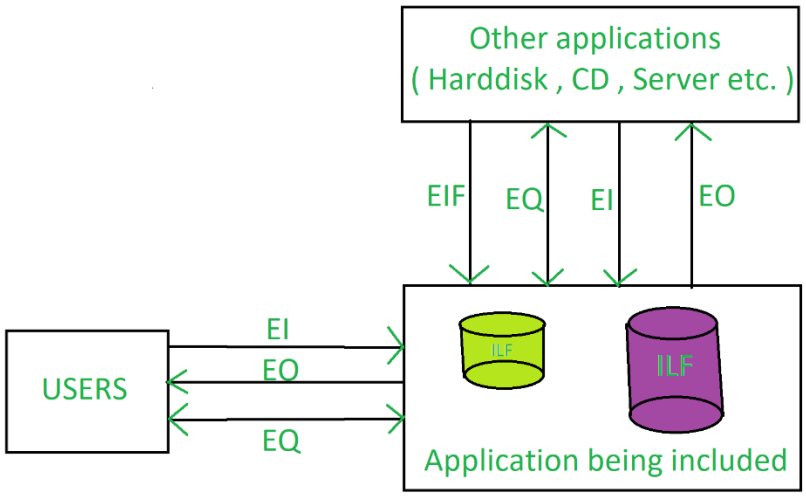
Internal Logical Files (ILF): Internal Logical Files that files controlled by the program. Each data file (or database table) is counted for ILF. There should be a approximately **10** database table which consist these information's: user id, user name, driver id, driver name, driver rating, payment id, payment method, user message id, user message, price etc.

External Interface Files (EIF): External Interface Files that files controlled by other programs. All machine-readable interfaces (import/export data file) that are used to transmit information to another system are counted as EIF. Map information are necessary for navigation and that can be exported as a machine-readable interface and also payment can be done with external interface. Then it can be counted as **2**.

External Input (EI): Each user input (screens, forms, dialog boxes, controls etc.) that provides distinct data to the software is counted as EI. In my project user and driver properties can counted as EI. Number of different user inputs is about **12** including user id, user name, user password, starting point of travel, end point of travel, credit card information's, driver id, driver name, messages from user and driver etc.

External Output (EO): Each user output that provides information to the user is counted AS EO. In this context, output refers to reports, screens, graphs, error messages, etc. In my project there should be a map to monitor drivers movement and also some messages to show the user the booking is accepted or not. There is also registration page, drivers comment/rating page, messaging page, payment page etc. It can be said there would be a approximately **16** EO.

External Query (EQ): An inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output. Things like nearest driver search, cheapest offer search, best rating driver search can be considered as a EQ. It can be said that there is **3** EQ.



Function Units	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Table Coefficient of Function Units

Type of Component	Complexity of Components			
	Low	Average	High	Total
External Input (EI)	8	2	2	44
External Output (EO)	6	6	4	82
External Query (EQ)	0	3	0	12
Internal Logical Files (ILF)	6	4	0	82
External Interface Files (EIF)	0	1	1	17
			Unadjusted Function Points	**237**

$$Adjusted\ FP = \ Unadjusted\ FP * VAF$$

$$VAF = (\sum_{i=1}^{14} GSC_i * 0.01) + 0.65$$

For calculating VAF:

Data communications: How many communication facilities are there to aid in the transfer or exchange of information with the application or system?

Distributed data processing: How are distributed data and processing functions handled?

Performance: Did the user require response time or throughput?

Heavily used configuration: How heavily used is the current hardware platform where the application will be executed?

Transaction rate: How frequently are transactions executed; daily, weekly, monthly, etc.?

On Line data entry: What percentage of the information is entered On Line?

End user efficiency: Was the application designed for end user efficiency?

On Line update: How many ILFs are updated by On Line transaction?

Complex processing: Does the application have extensive logical or mathematical processing?

Reusability: Was the application developed to meet one or many user's needs?

Installation ease: How difficult is conversion and installation?

Operational ease: How effective and/or automated are start up, back up, and recovery procedures?

Multiple sites: Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?

Facilitate change: Was the application specifically designed, developed, and supported to facilitate change?

Description	Rating
No influence	0
Incidental	1
Moderate	2
Average	3
Significant	4
Essential	5

According to table above, ratings would give for these 14 criteria.

	Rating
Data Communication	4
Distributed data processing	3
Performance	2
Heavily used configuration	3
Transaction rate	5
Online data entry	2
End user efficiency	5
Online update	3
Complex processing	4
Reusability	5
Installation ease	2
Operational ease	2
Multiple sites	2
Facilitate change	3
Total	**45**

$$VAF = (\sum_{i=1}^{14} GSC_i * 0.01) + 0.65 = 1.1$$

$$Adjusted\ FP = \ Unadjusted\ FP * VAF$$

$$Adjusted\ FP = \ 237 \ * 1.1 = 260.7$$

Calculating Size from Function Points:

The average ratio of FP/LOC is change according to the programming language. From Function Point Languages Table for java/ kotlin it can be determined as 53. (https://www.qsm.com/resources/function-point-languages-table)

$$LOC/\ FP = \ 53$$

$$LOC = 53 * 260.7 = 13817 \ \approx 14 \ KLOC$$

The Intermediate Cocomo formula:

$$E = a_i(KLOC)^{b_i}(EAF)$$

From table:

Software project	a_i	b_i	c_i
Organic	3.2	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	2.8	1.20	0.32

$$a_i = 3.0, b_i = 1.12, c_i = 0.35$$

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

When we multiply all marked values, we would find EAF as **1.39.**

$$E = a_i(KLOC)^{b_i}(EAF)$$

$$E = 3(14)^{1.12}(1.39) = 80.14\,pm$$

Development time:

$$D = 2.5(E)^{c_i} = 2.5(80.14)^{0.35} = 11.59\,m$$

Person required:

$$P = \frac{\mathrm{E}}{\mathrm{D}} = 6.9 \approx 7\,p$$

At least 7 person should work in this project.

4. RESOURCES

In estimation chapter the required person calculated as 7 people. There would be a **hierarchical** team and all members report to **one manager**.

4.1 Team Structure

Software Development Manager

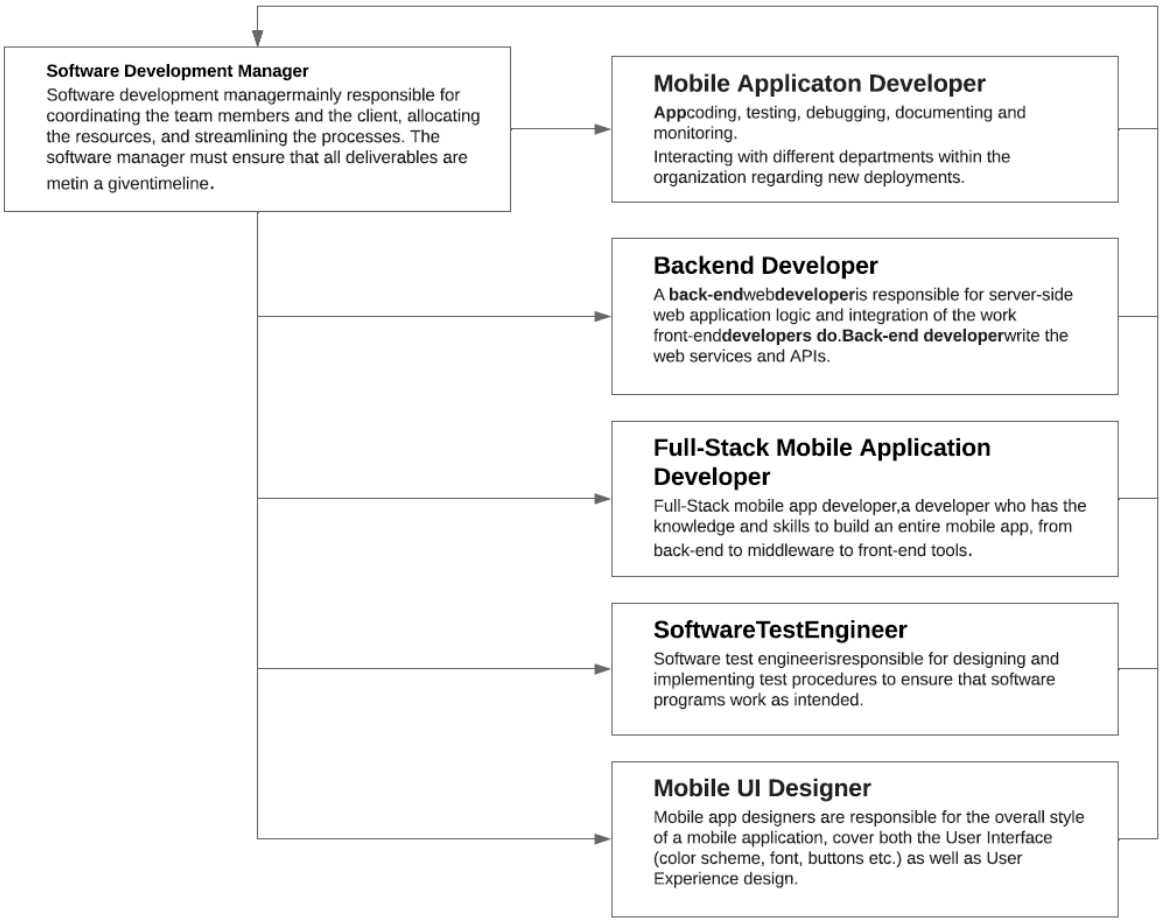
Backend Developer

Mobile Application Developer x2

Full-Stack Mobile Application Developer

Mobile UI designer

Software Test Engineer



PROJECT PLAN (Work Break-Down Structure)	Responsible Team Member
1. The customer can request a car via the app, either instantly or scheduled. Drivers should answer requests. **(XL)**	Backend Developer + Software Development Manager + Software Test Engineer
2. The driver has the option to accept or reject the booking. If a driver refuses a booking, then the next available driver may approve the request. **(S)**	Mobile UI designer + Backend Developer
3. The passenger should track the car and check out the estimated time of arrival. **(M)**	Mobile UI designer + Backend Developer
4. The app helps to calculate the price beforehand, and the customer can pay with their credit card. **(M)**	Backend Developer + Full-Stack Mobile Application Developer
5. Passengers can make comment to driver and give them a rating as well. **(S)**	Mobile UI designer
6. Messaging should be done between passenger and driver. **(S)**	Full-Stack Mobile Application Developer + Mobile Application Developer
7. There would be a separate UI and logic for both drivers and passenger, there would be 2 registration option: Driver and Passenger. **(M)**	Backend Developer + Mobile UI designer
8. Driver should monitor nearest passengers and make a booking offer to them. **(M)**	Full-Stack Mobile Application Developer + Backend Developer + Mobile Application Developer

4.2 Some Software Tool and Services Used

- AWS
- Apache JMeter for testing
- MySQL
- IntelliJ IDEA (IDE)
- PHP

4.3 Hardware Properties

- Minimum 2 GHz CPU
- Minimum 4GB RAM
- Minimum 5GB Hard-Disk

5. SCHEDULE

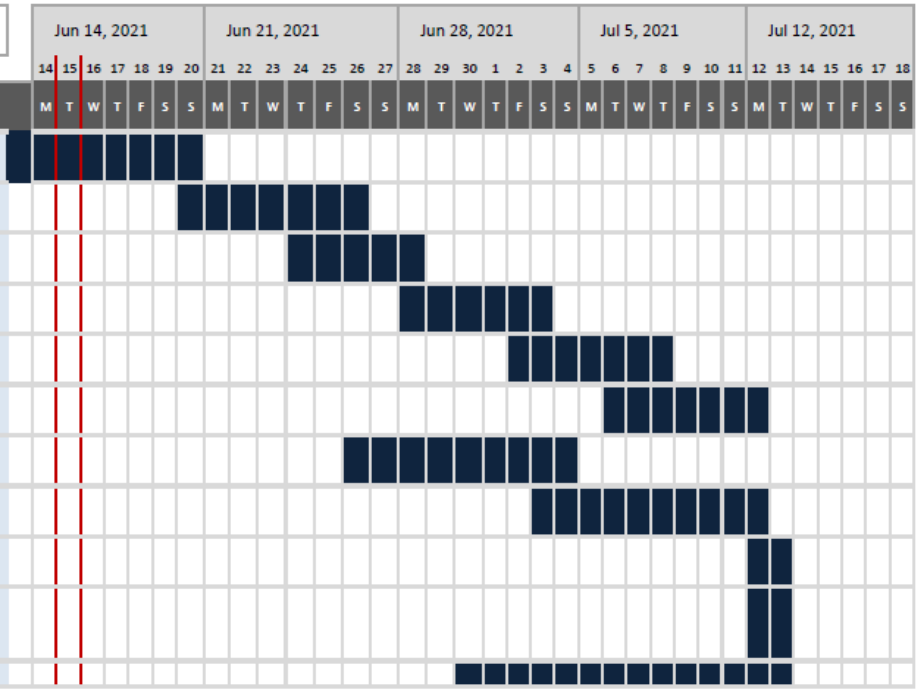
The project tasks and dates are shown in table below.

Taxi Booking Application

Project Start: Tue, 6/15/2021

Display Week: 1

TASK	ASSIGNED TO	PROGRESS
Determine requirement and specifications	Software Development Manager	50%
Making Requirement Analysis	Software Development Manager	10%
Design (UI design, Software Architecture, Web Services)	Software Development Manager + Mobile Application Developer	20%
Database Design	Full Stack Mobile Application Developer	10%
Making Database Normalization	Full Stack Mobile Application Developer	0%
Backend Implemantation	Backend Developer + Full Stack Mobile Application Developer	0%
Finalize UI design (page layouts, screens, pictures)	Mobile UI Designer	0%
Frontend Implemantation	Full Stack Mobile Application Developer + Frontend Developer	0%
Verification (Install, testing and debugging)	Software Test Engineer + Mobile Application Developer	0%
Build 1.0 Main Functions	Mobile Application Developer + Full Stack Mobile Application Developer	0%
Documentation	Mobile Application Developer	0%



6. RISKS

Risks	Probability(%)	Impact
Staff turnover will be high	20	4
Staff inexperienced	20	3
Funding will be lost	20	2
Lack of training on tools	30	4
Unrealistic expectations from users	30	2
Larger number of users than planned	40	2
Unstable project requirements	40	2
Wrong distribution of work to the project team members	40	1
Delivery deadline will be tightened	50	2
Customer will change requirements	50	2
Unstable project scope	50	1
There would be a less driver compared the user's demand	60	1
Inadequate number of people on the project team	70	3
Product competition	70	2
Size estimate may be significantly low	70	2

Impact values:

1—catastrophic

2—critical

3—marginal

4—negligible

The overall risk exposure, RE, is determined using the following relationship [Hal98]:

$$RE = P \; x \; C$$

where P is the probability of occurrence for a risk, and C is the cost to the project should the risk occur.

Risk exposure can be computed for each risk in the risk table, once an estimate of the cost of the risk is made. The total risk exposure for all risks (above the cutoff in the risk table) can provide a means for adjusting the final cost estimate for a project. It can also be used to predict the probable increase in staff resources required at various points during the project schedule.

6.1 Risk Mitigation (Aversion)

Risks have to be identified, prioritized, mitigated, and tracked. All aspects of risk management are described in this section.

Risks	**Strategy for avoid risk**
Staff turnover will be high	Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market). / Assign a backup staff member for every critical technologist.
Staff inexperienced	Give educational courses to team members.
Lack of training on tools	Transfer money from budget for trainings.
Unrealistic expatations from users	Give short introduction about mobile application when users first use this application.
Larger number of users than planned	If there are so many users make a plan B and increase budget and personal immediately.
Unstable project requirements	When determining requirements try to be realistic and guess future changes.
Wrong distribution of work to the project team members	Make job distribution according to team members skill and experiences.
Delivery deadline will be tightened	In scheduling section, make detailed work break down structure and determine what effort needed and take consideration your team size.
Unstable project scope	In project plan document, scope should be exact. If scope is not stable cost should be so high compared to project budget.
There would be a less driver compared the user's demand	It is a little bit domain issue, but it is so crucial for application logic. If there is not enough driver take lower fee from driver and encourage them to use these application.
Inadequate number of people on the project team	When making estimation, try to determine realistic function points and maybe take team members opinion about job and the date they should have.
Product competition	There is so many application similar to these project, but this application is restricted in special area and because project budget is low, for both users and drivers this application should be cheaper.
Size estimate may be significantly low	Make size calculations, if there is doubt ask a consultant who is experienced about developing similar applications.