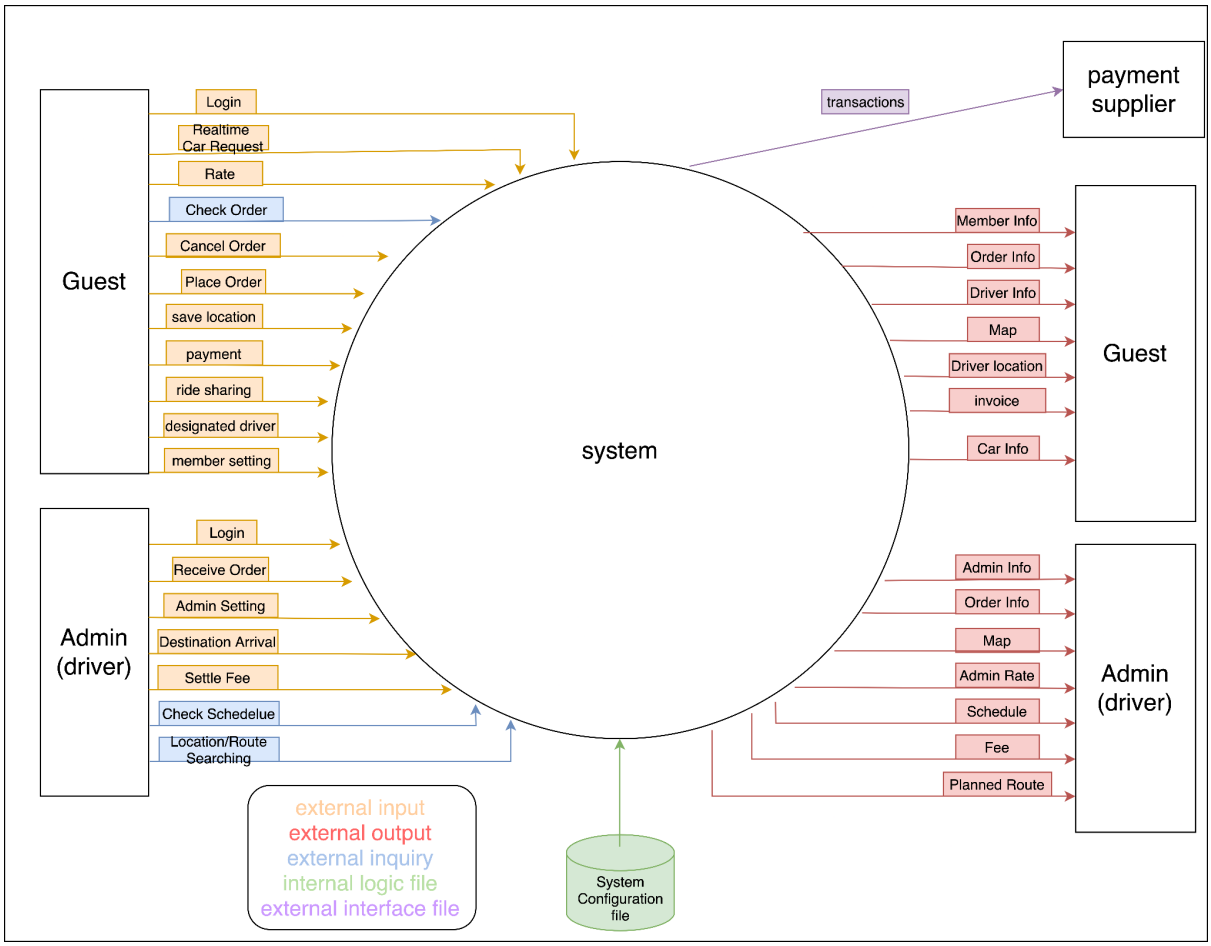
1.



系統基本功能：

- 乘車派遣服務平台的會員登入 ：

  使用者可以通過帳號密碼進行登入，司機同樣要進行會員登入。

- 叫車 (選擇車型、起點、終點、付款方式) ：

  選擇要叫的車種，路程的起點終點，並在最後決定付款方式。

- 司機詳細資料 (車型、車牌號碼、評價)：

  使用者可以查看司機的詳細資料，包含此司機評價，車種以及車牌號碼。

- 評價司機：

  使用者在結束搭乘後，可以為司機進行評分，舉例來說，可以採取一到五顆星的回饋。

- 查詢訂單細節(訂單內容、司機位置、預計到達時間) ：

使用者可以查訊訂單的下訂時間，司機目前所在位置、以及目前預計抵達上車地點的時間

- 司機 - 接收訂單、到達目的地、結算費用：

  司機可以藉由系統接收訂單、回報系統「已經抵達目的地」、並在抵達目的地後按下「費用結算」

系統額外功能：

- 儲存地點：

  使用者可以儲存自己慣用的目的地，以便下次使用時快速設定

- 代駕：

  提供給有飲酒，或因其他原因不適合自行駕車的客戶，替客戶將車開回指定地點，使用者需要提供車的資訊與需要代駕的啟程地與目的地。

- 共乘：輸入搭乘起點、終點及時間，開啟共乘功能的客戶可以加入相近路線的共乘，或是讓其他人加入。系統會通知使用者可以到哪邊加入別人的共乘，也會確認使用者是否願意接受其他人共乘的意願。

2.

F1 Reliable back-up and recovery: 1

F2 Data communication: 4

F3 Distributed functions: 1

F4 Performance: 5

F5 Heavily used configuration: 1

F6 Online data entry: 0

F7 Operational ease: 5

F8 Online update: 5

F9 Complex interface: 2

F10 Complex processing: 1

F11 Reusability: 3

F12 Installation ease: 0

F13 Multiple sites: 1

F14 Facilitate change: 3

TCF = 0.65 + 0.01 * 32 = 0.97

3.

EIs: 15*3 = 45

EOs: 14*4 = 56

EQs: 3*3 = 9

ILFs: 1*7 = 7

EIFs: 1*5 = 5

Count total = 122

4.

EIs: C++, 45* 64 = 2880

EOs: C++, 56* 64 = 3584

EQs: SQL, 9* 12 = 108

ILFs: JAVA, 7* 31 = 217

EIFs: C++, 5* 64 = 320

LOC = 7109

5.

(a)

RELY: 1.00 (Nominal)

DATA: 1.00 (Nominal)

CPLX: 0.85 (Low)

TIME: 1.11 (High)

STOR: 1.00 (Nominal)

VIRT: 1.15 (High)

TURN: 1.07 (High)

ACAP: 1.19 (Low)

AEXP: 0.91 (High)

PCAP: 0.86 (High)

VEXP: 1.10 (Low)

LEXP: 1.00 (Nominal)

MODP: 1.10 (Low)

TOOL: 1.10 (Low)

SCED: 1.00 (Nominal)

EAF = 1.44

(b)

E = 3.0 * (7.109)^ 1.12 * 1.44 = 38.86 person-months.

duration = 2.5 * (38.86)^ 0.35 = 9.00 months.

6.

(a)

我們選擇Story Points估算方法，Story Points 是一個單位，團隊會將功能分解為User Stories，並根據User Stories的大小、複雜度和風險等因素來估算Story Points，可使用 Fibonacci 數列或 T-Shirt 大小等方式進行量化，如此就可以分析User Stories間的相對大小。

分析步驟：

1. 確定故事：先確定專案中的User Stories，並將其描述清楚。一個故事通常包括一個或多個使用者需求，描述使用者希望系統實現的功能。

2. 估算 Story Points：針對每個User Stories進行討論，團隊會共同根據故事的複雜度、風險、需求等因素將其估算成一個 Story Point 的值，通常會有一些參考指標，例如：簡單User Stories有1-3 個 Story Points，中等User Stories有5-8 個 Story Points，複雜User Stories有13-20 個 Story Points。

3. 計算總 Story Points：將每個故事的 Story Point 值相加，得出專案的總 Story Points ，這個值會被視為專案大小的衡量標準。

4. 根據 Story Points 估算時間：開發團隊可以根據過去的經驗和累積的資料，估算每個 Story Point 所需的時間和人力。例如，團隊可以估算每個 Story Point 需要 1 天的開發時間，這樣專案的總開發時間就可以根據總 Story Points 值來計算。

也可以透過迭代的方式來預估，通過實際完成的 Story Points 數量和所花費的時間進行確認之前的預測是否準確，以此來更精準的來推估 Story Points 和時間之間的換算比。

Story Points是一種通用的敏捷專案管理方法，目前被國內外公司廣泛使用，使用公司有Microsoft, IBM, Amazon, Spotify等大型軟體公司。

(b)

以下針對我們的乘車派遣服務平台系統進行估算。

會員登入 ：3 簡單

叫車 (選擇車型、起點、終點、付款方式) ：15 複雜

司機詳細資料 (車型、車牌號碼、評價)：7 中等

評價司機：3 簡單

查詢訂單細節：3 簡單

司機 - 接收訂單、到達目的地、結算費用：7 中等

儲存地點：3 簡單

代駕：10 複雜

共乘：10 複雜

Total Story Points:61

Time per Story:5 天

估計專案完成時間：61*5天=305天＝大約10個月

估算的結果以工期上來說與FP 和 COCOMO都不同但相近，因為Story Points與 FP 和 COCOMO 相比，FP 和 COCOMO 把重點放在利用功能點和程式碼行數來估算專案大小，而 Story Points 則更注重故事的複雜度和風險等因素。因此，在使用 Story Points 進行估算時，團隊可能會更加重視需求分析和風險管理，因為這會對估算結果產生影響。

另外，敏捷方法重視互動、工作的運作、敏捷價值觀等等，較少專注在傳統的計算方式如 LOC 和 ECF 等。因此，在敏捷專案中，估算方法通常更加彈性以適應快速變化的需求和不斷發展的專案環境。

(c)

Story Points相較於FP 和 COCOMO等方法參雜了較多的主觀認知因此能夠將過去的專案經驗融合到複雜度和風險的預估中，因此對於敏捷開發等複雜專案的估算等有更佳的估算效果。除此之外，Story Point也無需仰賴程式碼行數等具體數據，因此評估更加快速且靈活，然而也因為估算結果主觀，若開發團隊較無經驗會導致一定程度的預估誤差，相較之下FP 和 COCOMO 則更加客觀，卻也可能無法確切表達團隊的實質預估狀況。