**ບົດທີ 1**

# **ບົດສະເໜີ**

## ຄວາມສຳຄັນຂອງບັນຫາ

ອີງໃສ່ເພື່ອຄວາມສະດວກສະບາຍຂອງປະຊາຊົນລາວເຮົາ ແລະ ຄົນຕ່າງປະເທດ, ນັກທ່ອງທ່ຽວ, ພະນັກງານ, ພໍ່ຄ້າຊາວຂາຍຕະຫຼອດຮອດພໍ່ແມ່ປະຊາຊົນທີ່ເດີນທາງເຂົ້າມາທ່ອງທ່ຽວ ຫຼື ມາເຮັດວຽກເຮັດງານທີ່ນະຄອນຫຼວງວຽງຈັນຂອງພວກເຮົາເພື່ອໃຫ້ການໄປມາສະຖານທີ່ຕ່າງໆພວກເຮົາຈຶ່ງໄດ້ສ້າງຕັ້ງສະຖານນີຂົນສົ່ງໂດຍສານນີ້ຂຶ້ນມາ.

ສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້ ເປັນບ່ອນບໍລິການຮັບ-ສົ່ງຜູ້ໂດຍສານ, ສິນຄ້າວັດຖຸສິ່ງຂອງ ແລະ ສັດ ຈາກຈຸດໜຶ່ງໄປຫາອີກຈຸດໜຶ່ງ ຊຶ່ງສະຖານນີຂົນສົ່ງໂດຍສານທາງໄກສາຍໃຕ້ນີ້ແມ່ນໄດ້ສ້າງ ຕັ້ງຂຶ້ນໃນວັນທີ 1 ກັນຍາ 2016 ເຊິ່ງມີລາຍລະອຽດດັ່ງລຸ່ມນີ້: ທີ່ຕັ້ງ ແລະ ພາລະບົດບາດ ຂອງສະ ຖານນີຂົນສົ່ງທາງໄກສາຍໃຕ້ແມ່ນສະຖານນີໜຶ່ງຊຶ່ງຕັ້ງຢູ່ ບ້ານ ສະພັງມຶກ, ເມືອງ ໄຊທານີ, ນະຄອນ ຫຼວງວຽງຈັນ, ຖະໜົນເລກທີ 450 ປີ ໃກ້ກັບ ສີ່ແຍກໄຟແດງດົງໂດກ. ສະຖານນີຂົນສົ່ງທາງໄກສາຍໃຕ້ ປະກອບມີຫຼາຍໜ່ວຍງານຄື: ອຳນວຍການໃຫ່ຍມີ 1 ທ່ານ, ເລຂານຸການມີ 1 ທ່ານ, ໜ່ວຍງານແຜນການມີ 1 ທ່ານ, ໜ່ວຍງານຮັບ-ຈ່າຍເງິນມີ 1 ທ່ານ, ໜ່ວຍງານບໍລິການຂາຍປີ້ມີ 7 ທ່ານ, ໜ່ວຍງານຮັກສາຄວາມປອດໄພມີ 6 ທ່ານ,ໜ່ວຍງານບໍລິການເຮືອນພັກມີ 8 ທ່ານ ແລະ ບັນດາບໍລິສັດ ທິ່ເຂົ້າມາດໍາເນີນທຸລະກິດ ໃນສະຖານຂົນສົ່ງໂດຍສານປະກອບມີ: ບໍລິສັດ ຂົນສົ່ງໂດຍສານຈິດປະສົງ ຍອດນິຍົມ, ບໍລິສັດ ແສງສົມບູນ ຂົນສົ່ງໂດຍສານ, ບໍລິສັດ ແສນສະບາຍ ຂົນສົ່ງໂດຍສານ, ບໍລິສັດ ຈັນທະຈອນ ຂົນສົ່ງໂດຍສານ, ບໍລິສັດ ຈໍາປາສັກ ຂົນສົ່ງໂດຍສານ, ບໍລິສັດ ແສງຈະເລີນ ລົດຕຽງນອນ, ບໍລິສັດ ກຽງໄກ VIP, ບໍລິສັດ ສີທອນ ພວງປະເສີດ ລົດຕຽງນອນ. ນອກຈາກນີ້ສະຖານນີຍັງ ມີສະຖານທີ່ພັກ , ຮ້ານຄ້າ, ຮ້ານຂາຍຍ່ອຍ, ຮ້ານອາຫານ ແລະ ສິ່ງອໍານວຍຄວາມສະດວກຕ່າງໆ ໄວ້ເພື່ອບໍລິການຜູ້ໂດຍສານທີ່ມາລໍຖ້າລົດໄປຈຸດໝາຍປາຍທາງ.​

## ຈຸດປະສົງຂອງການສຶກສາ

ວັດຖຸປະສົງຫຼັກຂອງບົດຈົບຊັ້ນໃນຄັ້ງນີ້ແມ່ນອີງໃສ່ໃນການພັດທະນາ ແລະ ເພີ່ມປະສິດທິພາບໃນການບໍລິການຂອງສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້, ພວກຂ້າພະເຈົ້າຈຶ່ງໄດ້ມີແນວຄິດທີ່ຈະສ້າງ ແລະ ພັດທະນາ ລະບົບຈອງປີ້ລົດເມສາຍໃຕ້ອອນລາຍຂຶ້ນມາເພື່ອເເກ້ໄຂບັນຫາໃຫ້ກັບທາງສະຖານນີຕາມວຽກຂອງເເຕ່ລະພະແນກເຖິງບັນຫາໃນການເຮັດວຽກ ດັ່ງນັ້ນ, ພວກຂ້າພະເຈົ້າຈຶ່ງໄດ້ກໍານົດເຫດຜົນຂອງການຄົ້ນຄ້ວາຕາມຈຸດປະສົງດັ່ງນີ້:

* ເພື່ອສຶກສາບັນຫາທີ່ເກີດຂຶ້ນໃນປະຈຸບັນ ແລະ ຄວາມຕ້ອງການຂອງລະບົບ.
* ເພື່ອສ້າງລະບົບຂາຍປີ້ລົດອອນໄລຂອງສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້.
* ເພື່ອສ້າງຮູບແບບການຈັດການຂໍ້ມູນການໃຫ້ບໍລິການ.
* ເພື່ອເຜີຍແຜ່ຂໍ້ມູນການຂາຍປີ້ລົດເມຂອງສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້.
* ເພື່ອການລາຍງານໃຫ້ສະດວກ ແລະ ຖືກຕ້ອງ.

## ຂອບເຂດການຄົ້ນຄວ້າ

ລະບົບຈອງປີ້ລົດເມສາຍໃຕ້ແບບອອນລາຍ ເປັນລະບົບແບບ Client-Server Web Application ເຊິ່ງປະກອບດ້ວຍໜ້າວຽກຫຼັກດັ່ງນີ້:

* ຈັດການຂໍ້ມູນພື້ນຖານ (ຂໍ້ມູນພະນັກງານ, ຂໍ້ມູນລົດ, ຂໍ້ມູນປະເພດລົດ, ຂໍ້ມູນສາຍທາງ)
* ສະໝັກສະມາຊິກ
* ບໍລິການ (ຈອງປີ້, ອອກປີ້)
* ລາຍງານ : (ລາຍງານການຈອງ, ລາຍງານຂໍ້ມູນພະນັກງານ, ລາຍງານຂໍ້ມູນສາຍທາງ, ລາຍງານຂໍ້ມູນລົດ, ລາຍງານຂໍ້ມູນຊຳລະເງິນ)

## ປະໂຫຍດທີ່ຄາດວ່າຈະໄດ້ຮັບ

ໃນການຂຽນບົດຈົບຊັ້ນໃນຄັ້ງນີ້ ຫຼັງຈາກສໍາເລັດໂຄງການນີ້ແລ້ວ Web Application ແມ່ນຕ້ອງໃຫ້ໄດ້ຮັບຜົນຕົວຈິງ ແລະ ສາມາດນໍາເຂົ້າມາໃຊ້ໃນວຽກງານຕົວຈິງໄດ້ຢ່າງແນ່ນອນ.

* ໄດ້ລະບົບຈອງປີ້ລົດເມແບບອອນລາຍຂອງສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້.
* ໄດ້ລະບົບທີ່ຈະຊ່ວຍແກ້ໄຂບັນຫາການຈອງໄດ້ສະດວກ ແລະ ວ່ອງໄວຂຶ້ນ.
* ໄດ້ລະບົບຊ່ວຍເພີ່ມຊ່ອງທາງໃນການຂາຍປີ້ໃຫ້ກັບຜູ້ປະກອບການ.
* ມີລະບົບເຜີຍແຜ່.
* ໄດ້ລະບົບທີ່ສາມາດສ້າງລາຍງານໄດ້ຢ່າງສະດວກ ແລະ ຖືກຕ້ອງ.

# **ບົດທີ 2**

# **ທົບທວນເອກະສານ ແລະ ບົດຄົ້ນຄວ້າທີ່ກ່ຽວຂ້ອງ**

## ທົບທວນທິດສະດີທີ່ກ່ຽວຂ້ອງ

### ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບຖານຂໍ້ມູນ

#### 2.1.1.1 ຄວາມໝາຍຂອງຖານຂໍ້ມູນ

Databases ຫຼື ຖານຂໍ້ມູນຄືກຸ່ມຂອງຂໍ້ມູນທີ່ຖືກເກັບລວບລວມໄວ້ໂດຍມີຄວາມສໍາພັນເຊິ່ງກັນ ແລະ ກັນໂດຍບໍ່ໄດ້ບັງຄັບວ່າຂໍ້ມູນທັງຫມົດນີ້ຈະຕ້ອງເກັບໄວ້ໃນແຟ້ມຂໍ້ມູນດຽວກັນ ຫຼື ແຍກເກັບຫຼາຍໆແຟ້ມຂໍ້ມູນ.

ລະບົບຖານຂໍ້ມູນຄືລະບົບທີ່ລວບລວມຂໍ້ມູນຕ່າງໆທີ່ກ່ຽວຂ້ອງກັນເຂົ້າໄວ້ດ້ວຍກັນຢ່າງມີລະບົບ,ມີຄວາມສໍາພັນລະຫວ່າງຂໍ້ມູນຕ່າງໆທີ່ຊັດເຈນ ໃນລະບົບຖານຂໍ້ມູນຈະປະກອບດ້ວຍແຟ້ມຂໍ້ມູນຫຼາຍແຟ້ມທີ່ມີຂໍ້ມູນກ່ຽວຂ້ອງກັນ, ສໍາພັນກັນເຂົ້າໄວ້ດ້ວຍກັນຢ່າງເປັນລະບົບ ແລະ ເປີດໂອກາດໃຫ້ຜູ້ໃຊ້ສາມາດໃຊ້ງານ ແລະ ຮັກສາປ້ອງກັນຂໍ້ມູນເຫຼົ່ານີ້ໄດ້ຢ່າງມີປະສິດທິພາບ ໂດຍມີຊອບແວຣ໌ທີ່ປຽບສະເຫມືອນຊື່ກາງລະຫວ່າງຜູ້ໃຊ້ ແລະ ໂປຣແກຣມຕ່າງໆທີ່ກ່ຽວຂ້ອງກັບການໃຊ້ຖານຂໍ້ມູນເອີ້ນວ່າລະບົບຈັດການຖານຂໍ້ມູນ ຫຼື DBMS (Databases Management System) ມີຫນ້າທີ່ຊ່ວຍໃຫ້ຜູ້ໃຊ້ເຂົ້າເຖິງຂໍ້ມູນໄດ້ງ່າຍສະດວກ ແລະ ມີປະສິດທິພາບການເຂົ້າເຖິງຂໍ້ມູນຂອງຜູ້ໃຊ້ອາດເປັນການສ້າງຖານຂໍ້ມູນ,ການແກ້ໄຂຖານຂໍ້ມູນ ຫຼື ການຕັ້ງຄໍາຖາມເພື່ອໃຫ້ໄດ້ຂໍ້ມູນມາໂດຍຜູ້ໃຊ້ບໍ່ຈໍາເປັນຕ້ອງຮັບຮູ້ກ່ຽວກັບລາຍລະອາດພາຍໃນໂຄງສ້າງຂອງຖານຂໍ້ມູນ.

#### 2.1.1.2 ຄຸນລັກສະນະຂອງຖານຂໍ້ມູນ

* ລົດຄວາມຊ້ຳຊ້ອນຂອງຖານຂໍ້ມູນໃຫ້ເຫຼືອນ້ອຍທີ່ສຸດ (Minimum Redundancy)
* ມີຄວາມຖືກຕ້ອງສູງສຸດ (Maximum Integrity)
* ມີຄວາມເປັນອິດສະຫຼະຂອງຂໍ້ມູນ (Data Independence)
* ມີລະບົບຄວາມປອດໄພຂອງຂໍ້ມູນສູງ (High Degree of Data Security)
* ການຄວບຄຸມຖານຂໍ້ມູນຈະຢູ່ສ່ວນກາງ (Logically Centralized Control)

#### 2.1.1.3 ຄວາມສໍາຄັນຂອງລະບົບຖານຂໍ້ມູນ

* ສາມາດລົດຄວາມຊ້ຳຊ້ອນຂອງຂໍ້ມູນໄດ້.
* ຫຼີກລ້ຽງຄວາມຂັດແຍ້ງຂອງຂໍ້ມູນໄດ້.
* ສາມາດໃຊ້ຂໍ້ມູນຮ່ວມກັນໄດ້.
* ສາມາດກໍານົດຄວາມເປັນມາດຕະຖານດຽວກັນຂອງຂໍ້ມູນໄດ້.
* ສາມາດກໍານົດລະບົບຄວາມປອດໄພຂອງຂໍ້ມູນໄດ້.

#### 2.1.1.4 ສະຖາປັດຕະຍະກໍາລະບົບຖານຂໍ້ມູນ

ສະຖາປັດຕະຍະກໍາລະບົບຖານຂໍ້ມູນ ມີ 3 ລະດັບຄື:

* ລະດັບພາຍໃນ (Internal Level)
* ລະດັບລັກສະນະແນວຄິດ (Conceptual Level)
* ລະດັບພາຍນອກ (External Level)

1. ລະດັບພາຍໃນ (Internal Level)

ເປັນການນໍາເອົາຂໍ້ມູນທີ່ໄດ້ຈາກລະດັບແນວຄິດມາວິເຄາະ ແລະ ອອກແບບໂດຍແປງໃຫ້ຢູ່ໃນຮູບແບບຂອງການຈັດການຖານຂໍ້ມູນ (DBMS) ທີ່ເລືອກໃຊ້ໂດຍເສີມແນວຄິດການເຮັດ Normalization car Demoralization.

1. ລະດັບລັກສະນະແນວຄິດ (Conceptual Level)

ເປັນການນໍາເອົາຂໍ້ມູນທີ່ໄດ້ຈາກການວິເຄາະຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ຂໍ້ມູນໃນລະດັບພາຍນອກມາອອກແບບຖານຂໍ້ມູນ ເພື່ອໄດ້ໂຄງຮ່າງຂອງຖານຂໍ້ມູນໃນລະດັບແນວຄິດທີ່ປະກອບດ້ວຍໂຄງສ້າງຂອງຖານຂໍ້ມູນສ່ວນໜຶ່ງເປັນຄວາມສໍາພັນກັນ.

1. ລະດັບພາຍນອກ (External Level)

ເປັນການນໍາເອົາຂໍ້ມູນທີ່ໄດ້ຈາກລະດັບພາຍໃນມາກໍານົດໂຄງສ້າງຂໍ້ມູນ ແລະ ການຈັດເກັບວິທີການເຂົ້າເຖິງການຈັດການດ້ານລະບົບຄວາມປອດໄພຂອງຂໍ້ມູນເພື່ອຖານຂໍ້ມູນເຮັດວຽກໄດ້ຢ່າງມີປະສິດທິພາບ.

### 2.1.2 ທິດສະດີທີ່ກ່ຽວຂ້ອງກັບ DFD (Data Flow Diagram)

#### 2.1.2.1 ຈຸດປະສົງຂອງ DFD

* ເປັນແຜ່ນພາບທີ່ສະຫຼຸບລວມຂໍ້ມູນທັງໝົດໄດ້ຈາກການວິເຄາະໃນລັກສະນະຂອງຮູບແບບທີ່ເປັນໂຄງສ້າງ.
* ເປັນຂໍ້ຕົກລົງຮ່ວມກັນລະຫວ່າງນັກວິເຄາະລະບົບ ແລະ ຜູ້ຊົມໃຊ້.
* ເປັນແຜ່ນພາບທີ່ໃຊ້ໃນການພັດທະນາຕໍ່ໃນຂັ້ນຕອນຂອງການອອກແບບ.
* ຮູ້ທີ່ໄປທີ່ມາຂອງຂໍ້ມູນທີ່ໄຫຼໃນຂະບວນການຕ່າງໆ.

#### 2.1.2.2 ສັນຍະລັກທີ່ໃຊ້ໃນແຜນວາດການໄຫຼຂໍ້ມູນ

|  |  |  |
| --- | --- | --- |
| **ຊື່** | **ສັນຍາລັກ** | **ຄວາມຫມາຍ** |
| Boundary Or External Entity |  | ຂອບເຂດໝາຍເຖິງພາກສ່ວນທີ່ກ່ຽວຂ້ອງກັບລະບົບ ເຊິ່ງລະບົບບໍ່ສາມາດຄວບຄຸມໄດ້ |
| Process |  | ປະມວນຜົນ ຫຼື ຫນ້າວຽກທີ່ເຮັດໃນໂຄງການນັ້ນໆ |
| Data Store |  | ບ່ອນຈັດເກັບຂໍ້ມູນ |
| Data Flow |  | ການໄຫຼຂອງຂໍ້ມູນ |
| Real-Time Link |  | ການເຊື່ອມໂຍງແບບໄກທີ່ມີການຕອບກັບແບບທັນທີທັນໃດ |

ຕາຕະລາງທີ 1: ສະແດງສັນຍາລັກ **Data Flow Diagram**

#### 2.1.2.3 ກົດຂອງ Process

* ຕ້ອງບໍ່ມີຂໍ້ມູນເຂົ້າພຽງຢ່າງດຽວ.
* ຕ້ອງບໍ່ມີຂໍ້ມູນອອກພຽງຢ່າງດຽວ.
* ຂໍ້ມູນທີ່ສົ່ງເຂົ້າຕ້ອງພຽງພໍໃນການສ້າງຖານຂໍ້ມູນທີ່ສົ່ງອອກ.
* ການຕັ້ງຊື່ Process ຕ້ອງໃຊ້ຄໍາກິລິຍາ (Verb) ເຊັ່ນ: ບັນທຶກຂໍ້ມູນ, ໃບບິນ, ກວດສອບຂໍ້ມູນລູກຄ້າ, ຈໍານວນເງິນເດືອນ.

#### 2.1.2.4 ຫຼັກການຂຽນແຜນວາດການໄຫຼຂໍ້ມູນ

1. Process:
   * ເມື່ອມີຂໍ້ມູນເຂົ້າໄປທີ່ Process ກໍ່ຕ້ອງມີຂໍ້ມູນ ຫຼື ຜົນຮັບອອກມາຈາກ Process ເຊັ່ນກັນຈະເປັນໄປບໍ່ໄດ້ທີ່ມີສະເພາະຂໍ້ມູນເຂົ້າຢ່າງດຽວ.
2. Data store:

* ຂໍ້ມູນຈະໄຫຼຈາກ Data store ໜຶ່ງໄປຫາ Data store ໜຶ່ງໂດຍກົງບໍ່ໄດ້ຈະຕ້ອງຜ່ານ Processເທົ່ານັ້ນ.
* ຂໍ້ມູນທີ່ສົ່ງຜ່ານ External entity ບໍ່ສາມາດໄຫຼເຂົ້າໄປ Data store ໂດຍກົງໄດ້ ຈະຕ້ອງໃຊ້ Process ເປັນຕົວກາງໃນການເຊື່ອມໂຍງເພື່ອຈັດເກັບຂໍ້ມູນໃນ Data store.
* ຂໍ້ມູນທີ່ໄຫຼຜ່ານຈາກ Data store ບໍ່ສາມາດເຊື່ອມໂຍງເຂົ້າກັບ External entity ໄດ້ໂດຍກົງຈະຕ້ອງຜ່ານ Process ເທົ່ານັ້ນ.

1. External entity:

* External entity ບໍ່ສາມາດເຊື່ອມໂຍງເຂົ້າຫາກັນໄດ້ຈະຕ້ອງໃຊ້ Process ເປັນຕົວກາງເພື່ອສົ່ງຜ່ານ ແລະ ຊື່ຂອງ External entity ຈະໃຊ້ຄໍານາມເທົ່ານັ້ນ.

1. Data flow:

* ການໄຫຼຂໍ້ມູນທີ່ມີຫົວຊີ້ໄປທີ່ Process ໝາຍເຖິງ Process ມີການອ່ານ ຫຼື ການດຶງຂໍ້ມູນຈາກ Data store ມາໃຊ້ວຽກ.
* ການໄຫຼຂໍ້ມູນຈາກ Process ທີ່ມີຫົວລູກສອນຊີ້ໄປຍັງ Data store ໝາຍເຖິງການ Update ຫຼື ການເພີ່ມຂໍ້ມູນລົງໄປທີ່ Data store.
* ການໄຫຼຂໍ້ມູນທີ່ມີຫົວລູກສອນທັງສອງດ້ານທີ່ເຊື່ອມໂຍງລະຫວ່າງ Process ກັບ Data store ໝາຍເຖິງມີການດຶງຂໍ້ມູນຈາກ Data store ມາປັບປຸງ ແລະ ມີການ Update ຂໍ້ມູນລົງໄປໃນ Data store.
* ການໄຫຼຂໍ້ມູນບໍ່ສາມາດຍ້ອນກັບໄປຍັງ Process ເດີມໄດ້ ຢ່າງໝ່ອຍຕ້ອງເຊື່ອມໂຍງຜ່ານProcess ໜຶ່ງເພື່ອສົ່ງຜ່ານຍ້ອນກັບມາຍັງ Process ເດີມ ແລະ ຊື່ທີ່ລະບຸໃນການໄຫຼຂໍ້ມູນຈະໃຊ້ຄໍານາມ.

|  |  |
| --- | --- |
| **ອະນຸຍາດ** | **ບໍ່ອະນຸຍາດ** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

ຕາຕະລາງທີ 2: ຮູບການປຽບທຽບແຜນວາດການໄຫຼຂໍ້ມູນທີ່ບໍ່ຖືກຕ້ອງ ແລະ ຖືກຕ້ອງ

### 2.1.3 ຄວາມໝາຍ ແລະ ສັນຍະລັກຂອງ Flowchart

ສັນຍະລັກ Flowchart ຄືຮູບພາບທີ່ໃຊ້ແທນຄວາມໝາຍການເຮັດວຽກງານໃນລັກສະນະຕ່າງໆ ພາຍໃນແຜນຜັງ (Flowchart) ປະກອບໄປດ້ວຍການເລີ່ມຕົ້ນ (Start), ການຈົບ (End), ການກະທຳ (Process), ການນຳເຂົ້າຂໍ້ມູນ (Input), ການສະແດງຜົນຂໍ້ມູນ (Output), ການຕັດສິນໃຈ (Decision), ຄຳອະທິບາຍ (Annotation), ຈຸດເຊື່ອມຕໍ່ (Connector), ທິດທາງການເຮັດວຽກງານ(Direction Flow).

ສັນຍະລັກເຫຼົ່ານີ້ເມື່ອຖືກນຳມາເຊື່ອມຕໍ່ກັນ ຈະກາຍເປັນ "ແຜນຜັງ (Flowchart)" ທີ່ສະແດງລຳດັບຂັ້ນຕອນການເຮັດວຽກງານເພື່ອ

* ເປັນ​ເຄື່ອງ​ມື​ໃນ​ການ​ຈັດ​ລຳ​ດັບ​ຄວາມ​ຄິດ.
* ເຫັນ​ລຳ​ດັບ​ຂັ້ນ​ຕອນ​ການ​ເຮັດ​ວຽກງານ​ທີ່​ຊັດເຈນ.
* **ສັນຍະລັກ Flowchart**

|  |  |
| --- | --- |
| ຮູບ​ພາບ​ສັນຍະລັກ | ຄວາມຫມາຍ​ຂອງ​ສັນຍະລັກ |
|  | ການ​ເລີ່ມ​ຕົ້ນ​ ຫຼື ຈົບ Flowchart (Start ຫຼື End) |
|  | ການ​ກະທຳ (Process) ຖືກ​ໃຊ້​ເພື່ອ​ສະແດງ​ທີ່​ການ​ກະທຳ​ໃນ Flowchart |
|  | ຮັບຂໍ້ມູນ |
|  | ການ​ຕັດ​ສິນ​ໃຈ (Decision) |
|  | ສະແດງຜົນທາງຈໍພາບ |
|  | ຈຸດ​ເຊື່ອມ​ຕໍ່ (Connector) |
|  | ທິດ​ທາງ​ການ​ເຮັດ​ວຽກງານ (Direction Flow) |

ຕາຕະລາງທີ 3: ສະແດງສັນຍາລັກ Flowchart

### 2.1.3 ແຜນວາດຄວາມສຳພັນຂອງ Entity (ER Diagram)

ການອອກແບບການສ້າງຕາຕະລາງຖານຂໍ້ມູນແບບຈໍາລອງ ER ແບ່ງອອກເປັນ 2 ຂັ້ນຕອນຫຼັກຄື: ຂັ້ນຕອນທໍາອິດເປັນການສ້າງແບບຈໍາລອງER ຂັ້ນຕອນນີ້ຈະກ່າວເຖິງຄວາມໝາຍຂອງສັນຍາລັກຕ່າງໆທີ່ໃຊ້ໃນແບບຈໍາລອງ ER, ຂັ້ນຕອນທີ 2 ແມ່ນການແປງແບບຈໍາລອງ ER ໃຫ້ເປັນຕາຕະລາງຂໍ້ມູນເພື່ອໃຊ້ເປັນຕົວຈັດການຂໍ້ມູນ. ເມື່ອເຂົ້າໃຈໃນທັງສອງແບບນີ້ແລ້ວສາມາດອອກແບບຖານຂໍ້ມູນ ແລະ ສ້າງຖານຂໍ້ມູນໃນແບບຈໍາລອງ ER ຂຶ້ນມາໃຊ້ງານດ້ວຍຕົວເອງ.

### ສັນຍາລັກທີ່ໃຊ້ໃນແບບຈໍາລອງ ER

ການອອກແບບຈໍາລອງຖານຂໍ້ມູນ ER ນັ້ນກ່ອນອື່ນຕ້ອງຮູ້ຈັກຄວາມໝາຍສັນຍາລັກຕ່າງໆທີ່ໃຊ້ໃນການອອກແບບຈໍາລອງ ER ໄດ້ຢ່າງຖືກຕ້ອງ ດັ່ງນັ້ນ ໃນຫົວຂໍ້ນີ້ຈະເວົ້າເຖິງຄວາມໝາຍ ແລະ ການໃຊ້ງານສັນຍາລັກຕ່າງໆຂອງ ER.

* 1. ເອັນຕີຕີ້ (Entity)

ເອັນຕີຕີ້ ຄືວັດຖຸທີ່ເຮົາສົນໃຈເຊິ່ງອາດເປັນໄດ້ທັງບຸກຄົນ, ສະຖານທີ່, ວັດຖຸ, ເຫດການ ຫຼື ແນວຄິດທີ່ກໍ່ໃຫ້ເກີດກຸ່ມຂອງຂໍ້ມູນທີ່ຕ້ອງການເອັນຕີຕີ້ແບ່ງອອກເປັນ 2 ປະເພດຄື:

* 1. Strong Entity:

ເປັນເອັນຕີຕີ້ທີ່ເກີດຂຶ້ນດ້ວຍຕົນເອງເປັນອິດສະຫຼະບໍ່ຂຶ້ນກັບເອັນຕີຕີໃດສັນຍາລັກທີ່ໃຊ້ຄືຮູບສີ່ຫຼ່ຽມ ແລະ ສາມາດເອີ້ນ Strong Entity ໄດ້ອີກຊື່ໜຶ່ງວ່າ Regular Entity.

**CUSTOMER**

**STOCK**

ຮູບທີ 1: ຮູບ Strong Entity

* 1. Weak Entity:

ເອັນຕີຕີ້ຊະນິດນີ້ຈະຂຶ້ນກັບເອັນຕີຕີ້ຊະນິດອື່ນໆບໍ່ສາມາດເກີດຂຶ້ນໄດ້ຕາມລໍາພັງ ແລະ ຈະຖືກລົບເມື່ອເອັນຕີຕີ້ຫຼັກຖືກລົບອອກ ສັນຍາລັກທີ່ໃຊ້ຄືຮູບສີ່ຫຼ່ຽມຊ້ອນກັນ.

**CUSTOMER**

**STOCK**

ຮູບທີ 2: ຮູບ Weak Entity

* 1. ແອັດທີຣບີວ(Attribute)

ແອັດທີຣບີວຄືຄຸນສົມບັດຂອງສົມບັດຂອງເອັນຕີຕີ້ສັນຍາລັກຂອງແອັດທີຣບີວຈະເປັນຮູບວົງຮີແອັດທີຣບີວໃດທີ່ຖືກໃຊ້ເປັນຄີຫຼັກຈະຖືກຂີດເສັ້ນກ້ອງກໍາກັບໄວ້.

emName

Position

emNo

**EMPLOYEE**

ຮູບທີ 3: ຮູບ Attribute

* 1. ຄວາມສໍາພັນ (Relation)

ຄວາມສໍາພັນໃນທີ່ນີ້ ໝາຍເຖິງຄວາມສໍາພັນລະຫວ່າງເອັນຕີຕີ້ ໂດຍແຕ່ລະຄວາມສໍາພັນຄວນມີຊື່ລະບຸໄວ້ເພື່ອໃຊ້ອະທິບາຍເຊິ່ງປົກກະຕິຈະໃຊ້ສັນຍາລັກຮູບດອກຈັນທີ່ພາຍໃນລະບຸຄໍາກິລິຍາໄວ້ເພື່ອອະທິບາຍຄວາມສໍາພັນ.

SUBJECT

teaches

LUCTURE

M

1

ຮູບທີ 4: ຮູບ Relation

### 2.1.5 ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບພາສາ NOSQL

ຫຼາຍ​ຄົນ​ຄົງ​ພໍ​ຈະ​ເຄີຍ​ໄດ້ຍິນ​ກັນ​ມາແດ່ກ່ຽວກັບ​ເທັກ​ໂນ​ໂລ​ຢີ​ ການ​ຈັດການ​ຂໍ້​ມູນ​ແບບ​ໃໝ່​ນີ້ ຊຶ່ງ​ກໍ​ຄື NoSQL ເມື່ອ​ເວົ້າ​ເຖິງ NoSQL ຈະ​ໄດ້ຍິນ​ຊື່​ເວັບ​ໄຊ​​ທີ່​ໃຫ່ຍ​ໆ ເຊັ່ນ Facebook, Twitter, FourSquare, Digg ແລະ ​ອື່ນ​ໆ.  ເຮັດໃຫ້​ເຮົາ​ຮັບ​ຮູ້​ວ່າ NoSQL ເປັນ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ສຳລັບ​ງານ​ທີ່​ຕ້ອງ​ຮອງ​ຮັບ​ຂໍ້​ມູນ​ຂະໜາດ​ໃຫ່ຍໆ ຮອງ​ຮັບ​ການ​ຂະຫຍາຍ​ລະບົບ​ໄດ້​ງ່າຍເປັນ​ຕົ້ນ.

ຊຶ່ງ​ກໍ​ເປັນ​ເຊັ່ນ​ນັ້ນ​ແທ້ ແຕ່​ວຽກ​ທີ່​ນ້ອຍ​ໆຈະ​ເຮັດ​ຢ່າງໃດໃຊ້​ງານ​ໄດ້​ບໍ່ ຄຸ້ມ​ຄ່າ​ທີ່​ຈະ​ນຳ NoSQL ມາ​ໃຊ້​ງານ​ຂະໜາດ​ນ້ອຍ​ ຫຼື ບໍ່ ຫຼື​ໃຊ້ Relational Database ກໍ​ພຽງ​ພໍ​ແລ້ວ ຄຳ​ຕອບ​ຄືຂຶ້ນ​ຢູ່​ກັບ​ລັກສະນະ​ໃນ​ການ​ນຳ​ມາ​ໃຊ້​ງານ ກ່ອນທີ່ຈະ​ຕອບ​ຄຳ​ຖາມວ່າ NoSQL ເປັນ​ຄຳ​ຕອບ​ຂອງ​ລະບົບ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຂອງ​ຫຼື ບໍ່ ລອງ​ພິຈາລະນາ​ຫົວຂໍ້​ຕ່າງ​ໆດັ່ງ​ຕໍ່ໄປ​ນີ້:

1. **ຜູ້ໃຊ້ນັບມື້ນັບຫຼາຍ (BigUsers)**

ຈະ​ເຫັນ​ໄດ້​ວ່າ​ໃນ​ຊ່ວງ​ເວລາ​ທີ່​ຜ່ານມາ​ນີ້ ແລະ ​ໃນ​ປັດ​ຈຸ​ບັນຜູ້​ທີ່​ໃຊ້​ງານ Internet ມີ​ແນວ​ໂນ້ມ​ຫຼາຍ​ຂຶ້ນ​ເລື້ອຍ​ໆ ບໍ່​ວ່າ​ຈະ​ໃຊ້​ງານ​ຜ່ານ Desktop PC ຫຼື Smartphone ຊຶ່ງ​ເທັກ​ໂນ​ໂລ​ຊີ​ຂອງ​ອຸປະກອນ (Devices) ມີ​​ຫຼາຍ​​ຂຶ້ນ ແລະ ​ໃຊ້​ງານ​ໄດ້​ງ່າຍ​ຂຶ້ນ.

ການ​ພັດທະນາ​ລະບົບ​​ສາມາດ​ຮອງ​ຮັບ​ປະລິມານ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ແຕ່​ລະ​ອຸປະກອນ (Devices) ເປັນ​ສິ່ງ​ໜຶ່ງ​ທີ່​ຕ້ອງ​ນຳ​ມາ​ພິຈາລະນາ  ແລະ ​ບໍ່​ພຽງ​ແຕ່​ຕ້ອງ​ຮອງ​ຮັບ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ຂອງ​ຜູ້​ໃຊ້​ໄດ້​ເທົ່າ​ນັ້ນ  ເຮົາ​ຕ້ອງ​ຮອງ​ຮັບ​ວິທີ​ການ​ປ້ອນ​ຂໍ້​ມູນ​ແບບ​ໃໝ່ຄືແຕ່​ກ່ອນ​ຜູ້​ຈັດການ​ເນື້ອ​ຫາ​ຕ່າງ​ໆ ຄື Web Master, Web Editor, ຜູ້​ເບິ່ງ​ແຍງ​ລະບົບເປັນ​ຕົ້ນ. ແຕ່​ປະຈຸບັນຜູ້​ທີ່​ປ້ອນ​ຂໍ້​ມູນ​ຄືຜູ້​ໃຊ້​ບໍລິການ (users) ໂດຍ​ກົງຜ່ານ​ອຸປະກອນ (Devices) ຕ່າງ​ໆ​ທີ່​ມີ​ຫຼາກ​ຫຼາຍ ແລະ ​ການ​ປ້ອນ​ຂໍ້​ມູນ​ກໍ​ງ່າຍ​ກວ່າ​ແຕ່​ກ່ອນ​ອີກດ້ວຍ.

ຍັງມີ​ປັດ​ໄຈ​ອື່ນ​ໆເຊັ່ນ ເທດ​ສະການ​ສຳຄັນ​ໆທີ່​ຄົນ​ຈະ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ຫຼາຍ​ເປັນ​ພິເສດ ຫຼື ​ຜູ້​ໃຊ້​ງານ​ທີ່​ບໍ່​ແມ່ນ​ແຕ່​ປະເທດ​ເຮົາ​ເທົ່າ​ນັ້ນ ເພາະ​ໂລກ​ອິນ​ເຕີເນັດ​ເຖິງ​ກັນ ອາດຈະ​ຕ້ອງ​ເບິ່ງ​ວ່າ​ລະບົບ​ເຮົາ​ມີ​ຜູ້​ເຂົ້າ​ໃຊ້​ງານ​ຈາກ​ຕ່າງ​ປະເທດ ຫຼື ທົ່ວ​ໂລກ ​ຫຼື ບໍ່​ອີກ​ດ້ວຍ.

ດັ່ງ​ນັ້ນ ​ເຮົາ​ຕ້ອງ​ກັບ​ມາ​​ທົບທວນ​ວິທີ​ການ​ຈັດການ​ຖານ​ຂໍ້​ມູນ​ແລ້ວ​ວ່າ  ບໍ່​ແມ່ນ​ແຕ່​ເຮັດໃຫ້​ຮອງ​ຮັບ​ກັບ​ການ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ຂອງ​ຜູ້​ໃຊ້​ບໍລິການ​ເທົ່າ​ນັ້ນ ແຕ່​ຕ້ອງ​ຮອງ​ຮັບ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ຫລາຍ​ຂຶ້ນ​ເລື້ອຍ​ໆໄດ້​ອີກ.

1. **ປະເພດຂໍ້ມູນຕ່າງໆ ແລະ ຂໍ້ມູນທີ່ຕ້ອງການຈັດເກັບຫຼາຍຂຶ້ນເລື້ອຍໆ (BigData)**

  ຈາກ​ຕົວ​ແປ​ຂອງ​ຜູ້​ໃຊ້​ງານ​ມີ​ຫຼາຍ​ຂຶ້ນ​ເລື້ອຍ​ໆ ອຸປະກອນ​ໃນ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ກໍ​ຫຼາກ​ຫຼາຍປະ​ເພດ​ຂອງ​ຂໍ້​ມູນ​ທີ່​ໄດ້​ຈາກ​ແຕ່​ລະ​ອຸປະກອນ​ກໍ​ຫຼາກ​ຫຼາຍ​ປະ​ເພດ ເຊັ່ນ ຂໍ້ຄວາມ, ຮູບ​ພາບ, ສຽງ, ວິ​ດີ​ໂອ, ຕຳ ແໜ່ງ​ສະຖານ​ທີ່ (GeoLocation) ແລະ ​ອື່ນ​ໆ ແລະ ​ການ​ປ້ອນ​ຂໍ້​ມູນ​ເຫຼົ່າ​ນີ້​ກໍ​ງ່າຍ​ແສນ​ງ່າຍ ເພາະ​ເທັກ​ໂນ​ໂລ​ຢີ​ຂອງ​ຮາ​ດ​ແວ​ຣ໌ ແລະ ​ຊອບແວຣ໌​​ມີ​ການ​ພັດທະນາ​ຫຼາຍ​ຂຶ້ນ​ເລື້ອຍ​ໆ ໃຊ້​ງານ​ງ່າຍ​ຂຶ້ນສະ​ດວກ​ຂຶ້ນວ່ອງໄວ​ຂຶ້ນ​ເລື້ອຍ​ໆ.

    ດັ່ງ​ນັ້ນ ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ລັ່ງໄຫຼ​ເຂົ້າ​ມາ​ຈາກ​ອຸປະກອນ​ຕ່າງ​ໆ​ເຫຼົ່າ​ນີ້ ເຮົາ​ອາດຈະ​ຕ້ອງ​ນຳ​ມາ​​ວິ​ເຄາະ​ພຶດຕິກຳ​ຂອງ​ຜູ້​ໃຊ້​ບໍລິການ ການ​ສົ່ງ​ເສີມ​ກາ​ນຕະຫຼາດ ເຮັດ​ຂໍ້​ມູນ​ການ​ຕັດ​ສິນ​ໃຈ​ຂອງ​ຜູ້​ບໍລິຫານ ຂໍ້​ມູນ​ລູກ​ຄ້າ​ສຳພັນ ແລະ ​ອື່ນ​ໆ​ອີກ​ຫຼວງ​ຫລາຍການ​ນຳ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ແບບ​ເດີມ(Relational Database) ອາດຈະ​ບໍ່​ເໝາະ​ກັບ​ລັກສະນະ​ງານ​ບາງຢ່າງ​ອີກ​ຕໍ່ໄປ.

1. **ເຕັກໂນໂລຢີຮາດແວຣ໌ໄດ້ປ່ຽນແປງລາຄາຖືກຫຼຸດລົງແຕ່ວ່າປະສິດທິພາບດີຂື້ນ (Cloud Computing).**

ເຮົາ​ອາດຈະ​ເລີ່ມ​ໄດ້ຍິນ​ຄຳ​ວ່າ Cloud Technology  ຊຶ່ງ​ກໍ​ມີ​ຫຼາຍ​ປະ​ເພດ ແຕ່​ໃນ​ນີ້ເຮົາ​ຈະ​ເວົ້າ​ເຖິງ​ໃນ​ແງ່​ຂອງ​ການ​ນຳ​ມາ​ໃຊ້​ງານໂດຍ​ປະຈຸບັນ​ຖ້າ​ໃຜ​ເຄີຍ​ໄດ້​ລອງ​ໃຊ້​ງານ EC2 ຂອງ Amazon ມາແດ່​ແລ້ວຈະ​ຮູ້​ຈັກ​ເປັນ​ຢ່າງ​ດີ​ວ່າການ​ຈະ​ມີ​ເຄື່ອງ Server ແຮງ​ໆ​ຈັກເຄື່ອງ ເປັນ​ເລື່ອງ​ທີ່​ງ່າຍ​ຫຼາຍ ຫຼື ການ​ຈະ​ມີ Server 10 ເຄື່ອງ ຫຼື 20 ເຄື່ອງ ນຳ​ມາ​ຕໍ່​ເປັນ Database Cluster ນັ້ນ​ງ່າຍ​ຫຼາຍ​ພຽງແຕ່​ຄລິກ​ສ້າງ Instance ບໍ່​ຈັກ​ເທື່ອ​ກໍ​ໄດ້ Server ມາ​ໃຊ້​ງານ​ແລ້ວ ແລະ​ ລາ​ຄາ​ກໍ​ຖືກ​ຫຼາຍ​ ຖ້າ​ເຮົາ​ບໍ່​ໃຊ້​ງານ​ແລ້ວກໍ​ຍົກ​ເລີກ​ການ​ໃຊ້​ງານ ​ແລະ​ ຄືນ​ກັບ​ໄປ​ໄດ້​ທັນ​ທີ ເມື່ອ​ທຽບ​ກັບ​ສະໄໝ​ກ່ອນ​ທີ່​ເຮົາ​ຕ້ອງ​ການ​ມີ Server ຈັກ 10 ເຄື່ອງ ຈະ​ຕ້ອງ​ລົງ​ທຶນ​ຊື້​ເຄື່ອງ​ມາ​ຫຼາຍ ຖ້າ​ເຊົາ​ໃຊ້​ງານ​ແລ້ວຕ້ອງ​ແບກ​ຮັບ​ພາລະ​ເຄື່ອງ Server ເຫຼົ່າ​ນີ້​ໄວ້​ຊຶ່ງ​ເປັນ​ຕົ້ນ​ທຶນ​ທີ່​ແພງ​ຫຼາຍ.​

ຈາກ​ທີ່​ກ່າວ​ມາ​ຂ້າງ​ເທິງ ເຮົາ​ບໍ່​ໄດ້​ເນັ້ນ​ຂໍ້​ດີ​ຂອງ EC2 ແຕ່​ຢ່າງ​ໃດແຕ່​ກຳ​ລັງ​ຈະ​ເນັ້ນ​ວ່າພາບ ລວມ​ຂອງ​ການ​ໃຊ້​ງານ Server ເລີ່ມ​ປ່ຽນ​ໄປຄືໃຊ້​ງານ​ໄດ້​ງ່າຍ​ຂຶ້ນ ລາ​ຄາ​ຖືກ​ລົງ ແຕ່​ປະ​ສິດ​ທິ​ພາບ​ດີ​ຂຶ້ນ ຊຶ່ງ​ເປັນ​ສິ່ງ​ທີ່​ສຳຄັນ​ໃນ​ການ​ນຳ​ມາ​ພິຈາລະນາ​ຂອງ​ເທັກ​ໂນ​ໂລ​ຢີ​ດ້ານ​ຖານ​ຂໍ້​ມູນ​ຄື ຖ້າ​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຖານ​ຂໍ້​ມູນ​ໃຫຍ່​ໆ ຫຼື ຮອງ​ຮັບ​ຜູ້​ໃຊ້​ງານ​ໄດ້​ເປັນ​ຈຳນວນ​ຫຼາຍ​ ການ​ຂະຫຍາຍ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ເປັນ​ເລື່ອງ​ທີ່​ງ່າຍ​ຂຶ້ນ ຊຶ່ງ​ເຮັດ​ໄດ້​ໂດຍ​ການ​ເອົາ Server ມາ​ຕໍ່​ໆ​ກັນ​ອອກ​ໄປ ຫຼື​ ເອີ້ນວ່າ​ການ​ຂະຫຍາຍ​ອອກ​ແນວ​ນອນ (Scale Out) ບໍ່​ແມ່ນ​ການ​ຂະຫຍາຍ​ລະບົບ​ຄືແຕ່​ກ່ອນ ຄື​ຂະຫຍາຍ​ອອກ​ແນວ​ຕັ້ງ (Scale Up) ແລະ​ ຕ້ອງ​ໃຊ້​ເຄື່ອງ Server ທີ່​ມີ​ປະ​ສິດ​ທິ​ພາບ​ສູງ​ ຊຶ່ງ​ຈະ​ມີ​ຕົ້ນ​ທຶນ​ທີ່​ແພງ​ກວ່າ​ການ​ຂະຫຍາຍ​ແບບ​ແນວ​ນອນຫລາຍ​.

     ດັ່ງ​ນັ້ນ ​ການ​ຂະຫຍາຍ​ລະບົບ​ທີ່​ຢູ່​ເທິງ​ພື້ນ​ຖານ​ຂອງ NoSQL ຄື​ຮອງ​ຮັບ​ການ​ຂະຫຍາຍ​ລະ ບົບ​ແບບ​ແນວ​ນອນ (Scale Out) ຊຶ່ງ​ຈະ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ໄປ​ເກັບ​ທີ່​ເຄື່ອງ Server ຫຼາຍ​ເຄື່ອງ ແລະ ​ໃຊ້​ເຄື່ອງ Server ທົ່ວ​​ໄປ​ທີ່​ເອີ້ນວ່າ (Commodity Server) ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ໃຊ້ Server ທີ່​ເອີ້ນວ່າ Enterprise Server ທີ່​ມີ​ລາ​ຄາ​ແພງ​ຕາມ Spec ທີ່​ສູງ​ຂຶ້ນ​ເລື້ອຍ​ໆ ແລະ​ ການ​ບໍລິຫານ​ຈັດການ​ກໍ​ຍາກ​ຂຶ້ນ​ອີກ​ດ້ວຍ.

1. **ບັນຫາ​ຂອງ Relational Database**

ຖ້າ​ໃຜ​ທີ່​ໃຊ້​ງານ Relation Database ທີ່​ຕ້ອງ​ການ​ຮອງ​ຮັບ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຂະໜາດ​ໃຫຍ່​ ຕ້ອງ​ຫຼີກລ້ຽງບໍ່ໄດ້ເລື່ອງ​ການເຮັດ Sharding ແລະ Distributed Cache ເພາະ​ເປັນ​ຕົວ​ຫຼັກ​ທີ່​ຕ້ອງ​ເຮັດ ເພື່ອ​ຂະຫຍາຍ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ຂອງ Relational Database ໃຫ້​ສາມາດ​ຮອງ​ຮັບ​ຂໍ້​ມູນ​ທີ່​ຫລາຍ​ຂຶ້ນ ແລະ​ ຮອງ​ຮັບ​ຈຳນວນ​ການ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ລະບົບ​ໄດ້​ຫຼາຍ​ຂຶ້ນ.

1. **Manual Sharding**

    ການ​ແບ່ງ​ຕາ​ຕະລາງ​ຖານ​ຂໍ້​ມູນ (Table) ອອກ​ເປັນ​ສ່ວນ​ ແລ້ວ​ກໍ່​ກະ​ຈາຍ​ໄປ​ຈັດ​ເກັບ​ໃນ​ຫຼາຍ​ໆ Server  ເພື່ອ​ໃຫ້​ແຕ່​ລະ​ຕາ​ຕະລາງ (Table) ຂອງ​ຖານ​ຂໍ້​ມູນ​ບໍ່​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ຫຼາຍ​ເກີນ​ໄປ ເພາະ​ຖ້າ​ຂໍ້​ມູນ​ໃນ​ແຕ່​ລະ​ຖານ​ຂໍ້​ມູນ​ຫູາຍ​ເກີນ​ໄປ ຈະ​ເຮັດໃຫ້​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ຊ້າ​ ແຕ່​ບັນຫາ​ກໍ​ຈະ​ຕາມ​ມາ​ອີກ​ຄື ເມື່ອ​ຕ້ອງ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ອອກ​ໄປ​ໃນ​ແຕ່​ລະ Server ການ​ຈະ​ ເ​ກັບ​ຂໍ້​ມູນ ເຊັ່ນ: ເພີ່ມ, ແກ້​ໄຂ, ລົບ, ດຶງ​ຂໍ້​ມູນ​ມາ​ສະແດງ​ຕ່າງ​ໆ ຈະ​ຕ້ອງ​ເຮັດ​ຜ່ານ Application ຫຼື ​ຕ້ອງ​ມີ Server ບາງ​ໂຕ​ທີ່​​ຖ້າ​ດຶງ​ຂໍ້​ມູນ​ແຕ່​ລະ Server ມາ​ທັງໝົດ​ເປັນ​ກ້ອນ​ດຽວ ນັ້ນ​ໝາຍ​ຄວາມ​ວ່າ ເຮົາ​ຕ້ອງ​ເຮັດ​ດ້ວຍ​ໂຕ​ເຮົາ​ເອງ ບໍ່​ແມ່ນ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ຈັດການ​ໃຫ້ (Manual Sharding).

1. **Distributed Cache**

ເມື່ອ​ເຮົາ​ຕ້ອງ​ການ​ໃຫ້​ລະບົບ​ຮອງ​ຮັບ​ການ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ຫຼາຍ​​ໄດ້​ນັ້ນ ຖ້າ​ຈະ​ຕ້ອງ​ເຂົ້າ​ມາ​ອ່ານ​ຂໍ້​ມູນ​ຜ່ານ Database ໂດຍ​ກົງ​ມັນອາດຈະ​ຮອງ​ຮັບ​ບໍ່​ໄຫວ ຫຼື ​ເຮັດ​ໄດ້​ຊ້າ ດັ່ງ​ນັ້ນ ​ຈະ​ຕ້ອງ​ມີ​ການເຮັດ Cache Layer ຂຶ້ນ​ມາ ຄື​ແທນ​ທີ່​ຈະ​ເຂົ້າໄປ​ອ່ານ​ຈາກ​ຖານ​ຂໍ້​ມູນ​ໂດຍ​ກົງ ກໍ​ໃຫ້​ອ່ານ​ຜ່ານ Cache ກ່ອນ ດັ່ງ​ນັ້ນ​ການ​ອ່ານ​ຂໍ້​ມູນ​ຈາກ Cache ເປັນ​ການ​ອ່ານ​ຈາກ Memory ໂດຍ​ກົງ ເຮັດໃຫ້​ຮອງ​ຮັບ​ປະລິມານ​ການ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ໄດ້​ຫຼາຍ​ຂຶ້ນ.

    ແຕ່​ບັນຫາ​ຄືການເຮັດ Cache Layer ນີ້ຮອງ​ຮັບ​ສະເພາະ​ການ​ອ່ານ​ຂໍ້​ມູນ​ເທົ່າ​ນັ້ນ ບໍ່​ຮອງ​ຮັບ​ການ​ຂຽນ​ຂໍ້​ມູນ​ໄດ້ ຖ້າ​ຕ້ອງ​ການ​ຮອງ​ຮັບ​ການ​ຂຽນ​ຂໍ້​ມູນ​ປະລິມານ​ຫຼາຍ​ ແລະ​ ອ່ານ​ຂໍ້​ມູນ​ປະລິມານ​ຫຼາຍ​ ຈຶ່ງ​ເປັນ​ສິ່ງ​ທີ່ Relational Database ບໍ່​ສາມາດ​ຮອງ​ຮັບ​ງານ​ໃນ​ລັກສະນະ ອ່ານ​, ​ຂຽນ ຂໍ້​ມູນ​ປະລິມານ​ຫຼາຍ​ໄດ້​ດີ​ ແລະ​ ສິ່ງ​ສຳຄັນການເຮັດ Cache Layer ຈະ​ຕ້ອງ​ມີ​ການ​ດູແລຮັກສາ ແລະ​ ໃຊ້ Server ແຍກ​ອອກ​ໄປ​ຕ່າງ​ຫາກ​ອີກ​.

    ຈາກ​ຈຸດ​ນີ້​ເອງ​ການ​ເຮັດ Sharding ແລະ Caching ເປັນ​ສິ່ງ​ທີ່​ຖືກ​ພັດທະນາ​ຂຶ້ນ​ໃນ NoSQL ເທັ​​ກ​ໂນ​ໂລ​​ຢີ ໂດຍ​ຮອງ​ຮັບ Auto-Sharding ແລະ Integrated Caching ໃນ​​ຕົວ​ເອງ ດັ່ງນັ້ນ ​ເຮົາ​ຈຶ່ງ​ໄດ້​ເຫັນ NoSQL ຖືກ​ນຳໄປ​ໃຊ້​ງານ​ກັບ​ລະບົບ​ໃຫຍ່ໆເຊັ່ນ: Facebook, Twitter, FourSquare, Digg ແລະ ​ອື່ນໆ ເພ​​າະວ່າ NoSQL ອອກ​ແບບ​ມາ​ເພື່ອ​ຮອງ​ຮັບ​ຄວາມ​ຕ້ອງການ​ງານ​ໃຫຍ່ໆ​ໄດ້​ດີ​ໂດຍ​ສະເພາະ​ຢູ່​ແລ້ວ ແຕ່​ເຖິງ​ຢ່າງ​ໃດກໍ​ຕາມ​ຍັງ​ມີ​ຄຸນສົມບັດ​ອື່ນໆ ທີ່​ໜ້າ​ສົນໃຈ​ໃນ NoSQL ເທັ​​ກ​ໂນ​ໂລ​​ຢີ.

**ກ. ຄຸນສົມບັດ​ຂອງ NoSQL Database**

* **Dynamic Schemas**

    ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຕ່າງ​ໆໃນ​ຖານ​ຂໍ້​ມູນ​ແບບ Relational Database ເຮົາ​ຈະ​ຕ້ອງ​ມີ​ການສ້າງ Schema ຫຼື ຮູບ​ແບບ​ຂອງ​ໂຄງ​ສ້າງ​ຕາ​ຕະລາງວ່າ​ຈະ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຫັຍງ​ ເມື່ອ​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ເພິ່ມເຕີມ​ຕ້ອງ​ປ່ຽນ Schema ພາຍ​ຫຼັງ (Alter-Table) ກ່ອນ​ຈະ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຮູບ​ແບບ​ໃໝ່​ໄດ້.

    ແຕ່​ໃນ​ປະຈຸບັນການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ມີ​ການ​ປ່ຽນ​ແປງ​ຕະຫຼອດ​ເວລາ ເພາະ​ຄວາມ​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຕ່າງ​ໆມີ​ຫຼາກ​ຫຼາຍ​​ຂຶ້ນ​ເລື້ອຍ​ໆ ການ​ກຳນົດ​ໂຄງ​ສ້າງ​ຂອງ​ຕາ​ຕະລາງ​ຖານ​ຂໍ້​ມູນ ຫຼື ການ​ຕ້ອງ​ປ່ຽນ​ໂຄງ​ສ້າງ​ຖານ​ຂໍ້​ມູນ​ເລື້ອຍໆ ໂດຍ​ທີ່​ຂໍ້​ມູນ​ຍັງມີ​ຢູ່​ແລ້ວເປັນ​ເລື່ອງ​ທີ່​ຍາກ​ຫຼາຍ ​ຫຼື ເຮັດ​ບໍ່​ໄດ້​ເລີຍ ວິທີ​ການ​ຄື​ອາດຈະ​ຕ້ອງ​ແຍກ​ອອກ​ເປັນ​ຕາ​ຕະລາງ​ໃໝ່ ຊຶ່ງ​ເປັນ​ວິທີ​ແກ້​ບັນຫາ​ຊົ່ວ​ຄາວ​ເທົ່າ​ນັ້ນ.

    ລະບົບ​ຖານ​ຂໍ້​ມູນ​ແບບ NoSQL ເຮົາ​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ມີ Schema ທີ່​ຕາຍ​ຕົວ ຫຼື ​ບໍ່​ຕ້ອງ​ມີ Schema ກ່ອນທີ່ຈະ​ຈັດ​ເກັບ​ຂໍ້​ມູນ ຂໍ້​ມູນ​ແຕ່​ລະ​ແຖວສາມາດ​ຈັດ​ເກັບ​ໄດ້​ຕາມ​ຕ້ອງ​ການ ຈະ​ເພີ່ມ​ ຫຼື ​ຫຼຸດ ກໍ​ບໍ່​ມີ​ບັນຫາ​ກັບ​ລະບົບ ເຮັດໃຫ້​ເຮົາ​ສາມາດ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ໄດ້​ຕາມ​ທີ່​ຕ້ອງ​ການປ່ຽນ​ແປງ​ໄດ້​ຕະຫຼອດ​ເວລາ ສະ​ດວກ​ ແລະ ວ່ອງໄວ.

* **Auto-Sharding**

ເມື່ອ​ຂໍ້​ມູນ​ມີ​ຂະໜາດ​ໃຫຍ່ ຫຼື ​ເຮົາ​ຕ້ອງ​ການ​ເພີ່ມ​ປະ​ສິດ​ທິ​ພາບ​ການ ອ່ານ​ ແລະ ຂຽນ​ຂໍ້​ມູນ​ປະລິ ມານ​ຫຼາຍ​ ການເຮັດ Sharding ໃນ​ລະບົບ NoSQL Database ຈະ​ກໍ່ການ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ໄປ​ຫາ Server ​ອັດຕະ​ໂນ​ມັດ​​ (Auto-Sharding) ຜູ້​ພັດທະນາ (Developer) ບໍ່​ຕ້ອງ​ຂຽນ​ໂປຣ​ແກຣມ​ໃນ​ການ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ເອງຄືກັບ Relational Database.  
    ການ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ອອກ​ໄປ​ຫຼາຍ​ໆ Server ນີ້​ຍັງ​ເຮັດໃຫ້​ມີ​ຂໍ້​ດີ​ຄື ປະ​ຢັດ​ຕົ້ນ​ທຶນ​ໃນ​ການ​ຂະຫຍາຍ​ລະບົບ ເພາະ​ເປັນ​ການ​ຂະຫຍາຍ​ແບບ​ແນວ​ນອນ (Scale Out) ຊຶ່ງ​ສາມາດ​ນຳ Server ປົກກະຕິ​ທົ່ວ​ໄປ​ມາ​ໃຊ້​ງານ​ໄດ້ ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເປັນ Enterprise Server

* **Replication**

ການ​ສຳ​ເນົາ​ຂໍ້​ມູນ​ຈາກ​ເຄື່ອງ​ໜຶ່ງ​ໄປ​ອີກ​ເຄື່ອງ​ໜຶ່ງ (Replication) ເມື່ອ Server ໜຶ່ງ​ເສຍ​ຫາຍ ອີກ​ເຄື່ອງ​ໜຶ່ງ​ຈະ​ຂຶ້ນ​ມາ​ເຮັດວຽກ​ແທນ​ທັນ​ທີໂດຍ​ຂໍ້​ມູນ​ຂອງ​ແຕ່​ລະ​ເຄື່ອງ​ຈະ​ມີ​ຂໍ້​ມູນ​ຄືກັນດັ່ງ​ນັ້ນ Replication ເປັນ​ໜຶ່ງ​ຄຸນສົມບັດ​ທີ່​ຕອບສະໜອງ​ຕໍ່​ການ​ໃຊ້​ງານ​ທີ່​ຕ້ອງ​ການ​ຄວາມ​ຕໍ່​ເນື່ອງ​ໄດ້​ຕະຫຼອດ​ເວລາ (High Availability).

* **Integrated Caching**

ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ໃຊ້​ງານ​ເລື້ອຍໆ ເຂົ້າ​ໄວ້​ໃນ Memory (RAM) ຊຶ່ງ​ເປັນ​ຄຸນສົມບັດ​ເດັ່ນ​ຂອງ NoSQL ທີ່​ທັງຫມົດ Caching ໄວ້​ໃນ​ຕົວ​ເອງ​ຢູ່​ແລ້ວ ເຮົາ​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເຮັດ Cache Layer ຄືກັບ Relational Database ອີກ​ຕໍ່ໄປ ທີ່​ຕ້ອງ​ເຮັດ Cache Layer ແຍກ​ຕ່າງ​ຫາກ ​ແລະ​ ເບິ່ງແຍງຮັກສາ​ແຍກ​ອອກ​ໄປ​ຕ່າງ​ຫາກ​ອີກ​ດ້ວຍ.

**ຂ. ປະ​ເພດ​ຂອງ​ຖານ​ຂໍ້​ມູນ NoSQL**

NoSQL ຖືກ​ແບ່ງ​ປະ​ເພດ​ຕາມ​ລັກສະນະ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ແຕກ​ຕ່າງ​ກັນ ດັ່ງ​ນັ້ນ ​ການ​ຈະ​ເລືອກ NoSQL Database ໂຕໃດໂຕໜຶ່ງຈະ​ຕ້ອງ​ເບິ່ງ​ອີກ​ວ່າການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຂອງ​ຖານ​ຂໍ້​ມູນ​ເປັນ​ແບບໃດ ເຊັ່ນ:

- Document databases ເຊັ່ນ MongoDB, CouchDB, Elasticsearch

- Graph stores ເຊັ່ນ Neo4J, Infinite Graph, InfoGrid

- Key-value stores ເຊັ່ນ DynamoDB, Redis, MemcacheDB

- Wide-column stores ເຊັ່ນ Cassandra, Amazon SimpleDB, Hadoop / HBase

**ຄ. Open source License**

    ໂດຍ​ສ່ວນ​ໃຫຍ່​ແລ້ວ NoSQL ຈະ​ເປັນ​ລິຂະ​ສິດ​ແບບ Open source ຊຶ່ງ​ບໍ່​ຕ້ອງ​ເສຍ​ຄ່າ​ໃຊ້​ຈ່າຍ​ໃນ​ການ​ນຳ​ມາ​ໃຊ້​ງານ ດັ່ງ​ນັ້ນ ​ເຮົາ​ສາມາດ​ນຳ NoSQL Database ແຕ່​ລະ​ຕົວ​ມາ​ຕິດ​ຕັ້ງ​ໃຊ້​ງານ​ໄດ້​ໂດຍ​ບໍ່​ເສຍ​ຄ່າ​ໃຊ້​ຈ່າຍ​ໃດ​ໆ (ຟ​ຣີ).

**ງ. ນຳ NoSQL ມາ​ໃຊ້​ງານ​ຂະໜາດ​ນ້ອຍ​ໄດ້​ ຫຼື​ ບໍ່?**

    ຈາກ​ທີ່​ກ່າວ​ມາ​ແລ້ວ ຄົງ​ພໍ​ຈະ​ຕອບ​ຄຳ​ຖາມ​ນີ້​ໄດ້​ວ່າການ​ນຳ NoSQL Database ​ເມື່ອ​ນຳ​ມາ​ໃຊ້​ໃນ​ງານ​ຂະໜາດ​ໃຫຍ່​ນັ້ນເໝາະ​ສົມ​ຢ່າງ​ແນ່ນອນ ແຕ່​ຖ້າ​ເປັນ​ລະບົບ​ທົ່ວ​ໆ​ໄປ​ຄວນ​ຈະ​ນຳ NoSQL ມາ​ໃຊ້​ງານ ​ຫຼື ​ບໍ່

    ຄຳ​ຕອບ​ຄື ຂຶ້ນ​ຢູ່​ກັບ​ລັກສະນະ​ວຽກ​ວ່າເຮົາ​ຈະ​ໃຊ້​ຄຸນສົມບັດ​ຫັຍງ​ຂອງ NoSQL ຖ້າ​ເຮົາ​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ບໍ່​ຕ້ອງ​ຢຶດ​ຕິດ​ກັບ​ໂຄງ​ສ້າງ (Dynamic Schema) ແລະ ​ຕ້ອງ​ການ​ເຂົ້າ​ໃຊ້​ງານ**​**ລະບົບ​ທີ່​ວ່ອງໄວ (Integrated Caching) ຂໍ້​ມູນ​ອາດຈະ​ຢັງ​ບໍ່​ຫຼາຍ​ເທົ່າ​ໃດອາດຈະ​ໃຊ້ NoSQL ໄດ້​ຢ່າງ​ແນ່ນອນ

ແຕ່​ຖ້າ​ບໍ່​ຕ້ອງ​ການ​ໃຊ້​ງານ (Dynamic Schema) ບໍ່​ຕ້ອງ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ທີ່​ວ່ອງໄວ (Integrated Caching) ເພາະ​ໃຊ້ Relational Database ກໍ​ເຮັດ​ໄດ້​ດີ​ຢູ່​ແລ້ວ Database Server ກັບ Web Server ກໍ​ຢູ່​ທີ່ Server ດຽວ​ກັນ ຂໍ້​ມູນ​ບໍ່​ຫລາຍ​ນັ້ນ ບໍ່​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ເພີ່ມ​ຂະຫຍາຍ​ຂຶ້ນ​ທຸກ​ມື້​ໆ ຜູ້​ເຂົ້າ​ໃຊ້​ງານ​ກໍ​ບໍ່​ໄດ້​ຫລາຍ ເບິ່ງ​ແລ້ວ​ວ່າ​ລະບົບ​​ບໍ່​ມີ​ແນວ​ໂນ້ມ​ຈະ​ຕ້ອງ​ຂະຫຍາຍ​ລະບົບ​ໃນ​ອະນາຄົດ​ອັນ​ໃກ້ ເຈົ້າ​ສາມາດ​ໃຊ້​ງານ Relational Database ໄດ້​ດີ​ຢູ່​ແລ້ວຢ່າງ​ບໍ່​ມີ​ບັນຫາ.

### 2.1.6 ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບພາສາ JavaScript

ພາສາ JavaScript ຫຼື ​ຫຍໍ້ JS ເປັນ​ພາສາ​ຂຽນ​ໂປຣ​ແກຣມ​ທີ່​ຖືກ​ພັດທະນາ ​ແລະ ​ປະຕິບັດ​ຕາມ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ​ຂອງ ECMAScript, ພາສາ JavaScript ນັ້ນ​ເປັນ​ພາສາ​ລະ​ດັບ​ສູງ ຄອມ​ພາຍ​ໃນ​ຂະນະ​ທີ່​ໂປຣ​ແກຣມ​ຣັນ (JIT) ແລະ​ ເປັນ​ພາສາ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບ​ຫຼາຍ​ຂະ​ບວນ​ເຊັ່ນ: ການ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບ​ຂັ້ນ​ຕອນ, ການ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບ​ວັດ​ຖຸ, ພາສາ JavaScript ມີ​ໄວ​ຍະກອນທີ່​ຄື​ກັບ​ພາສາ C ໃຊ້​ວົງ​ເລັບ​ເພື່ອ​ກຳນົດ​ບ່ອກ​ຂອງ​ຄຳ​ສັ່ງ ນອກ​ຈາກ​ນີ້ JavaScript ຢັງ​ເປັນ​ພາສາ​ທີ່​ມີ​ປະ​ເພດ​ຂໍ້​ມູນ​ແບບ​ໄດ​ນາ​ມິກ (Dynamic)​ ເປັນ​ພາສາ​ແບບ Prototype-based ແລະ First-class function.

ພາສາ JavaScript ນັ້ນ​ຖື​ວ່າ​ເປັນ​ເທັກ​ໂນ​ໂລ​ຊີ​ຫຼັກ​ຂອງ​ການ​ພັດທະນາ​ເວັບ​ໄຊ​ (World Wide Web)ມັນ​ເຮັດໃຫ້​ໜ້າ​ເວັບ​ສາມາດ​ຕອບ​ໂຕ້​ກັບ​ຜູ້​ໃຊ້​ໄດ້​ໂດຍ​ທີ່​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ລີເຟດໜ້າ​ໃໝ່ (Dynamic website) ເວັບ​ໄຊ​ຈຳນວນ​ຫຼາຍ​ໃຊ້​ພາສາ JavaScript ສຳລັບ​ຄວບ​ຄຸມ​ການ​ເຮັດ​ວຽກງານ​ທີ່​ດ້ານ Client-side ນັ້ນ​ເຮັດໃຫ້​ເວັບ​ບ​ຣາ​ວ​ເຊີຕ່າງ​ໆ ມີ JavaScript engine ທີ່​ໃຊ້​ສຳລັບ​ປະ​ມວນ​ຜົນສະຄິບ​ຂອງ​ພາສາ JavaScript ທີ່​ຣັນ​ເທິງ​ເວັບ​ບຣາວ​ເຊີເນື່ອງ​ຈາກ​ພາສາ JavaScript ເປັນ​ພາສາ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບ​ຫຼາຍ​ຮູບແບບເຮັດໃຫ້​ມັນ​ຮອງ​ຮັບ​ການ​ຂຽນ​ໂປຣ​ແກຣມ​ທັງ​ແບບEvent-driven, Functional ແລະ ​ແບບ​ລຳ​ດັບ​ຂັ້ນ​ຕອນມັນ​ມີ​ Library (APIs) ສຳລັບ​ເຮັດວຽກ​ກັບ​ຂໍ້ຄວາມ, ວັນທີ, Regular expression ແລະ ​ໂຄງ​ສ້າງ​ຂໍ້​ມູນ​ພື້ນ​ຖານ​ຢ່າງ Array ແລະ Map ຫຼື ​ທັງ Document Object Model (DOM) ຊຶ່ງ​ເປັນ API ທີ່​ໂດຍ​ທັ່ວ​ໄປ​ແລ້ວ​ສາມາດ​ໄດ້​ເທິງ​ເວັບບຣາວເຊີ.

ຢ່າງໃດ​ກໍ​ຕາມຕົວ​ຂອງ​ພາສາ JavaScript ເອງ​ບໍ່​ໄດ້​ມີ​ຟັງ​ຊັນ​ສຳລັບອິນພຸດ/ເອົາພຸດ(I/O) ທີ່​ມາ​ກັບ​ພາສາເຊັ່ນ: ຟັງ​​ຊັນ​ກ່ຽວກັບ Network ວຽກກ່ຽວ​ກັບ​ໄຟ​ລ ຫຼື Library​ ກ່ຽວກັບ​ກຣາ​ບຟິກໂດຍ​ທົ່ວ​ໄປ​ແລ້ວ​ສິ່ງເຫຼົ່ານີ້​ຈະ​ຖືກ​ໃຫ້​ມາ​ໂດຍ Host environment (ສະພາບ​ແວດ​ລ້ອມ​ທີ່​ໃຊ້​ຣັນ​ພາສາ JavaScript) ເຊັ່ນ ເວັບ​​ເວັບບຣາວເຊີ ຫຼື Node.js ຊຶ່ງ​ຈະ​ແຕກ​ຕ່າງ​ກັນ​ອອກ​ໄປ ​ຕົວ​ຢ່າງ ​ເຊັ່ນ:ການ​ຮັບ​ຄ່າ​ໃນ​ເວັບ​​ເວັບບຣາວເຊີ​ຈະ​ຜ່ານ​ຟັງ​ຊັນ prompt ຊຶ່ງ​ເປັນ​ສ່ວນຫນຶ່ງ​ຂອງ Browser Object Model (BOM) ຫຼື ​ຮັບ​ຄ່າ​ຈາກ HTML ຟອມຊຶ່ງ​ເປັນ​ສ່ວນຫນຶ່ງ​ຂອງ Document Object Model (DOM) ໃນຂະນະ​ທີ່​ເທິງ Node.js ເຮົາ​ສາມາດ​ຮັບ​ຄ່າ​ໄດ້​ຈາກ Input/Output Stream ຂອງ Command line ເຖິງ​ແມ່ນ​ວ່າ​ມັນ​ຈະ​ມີ​ຄວາມ​ຄ້າຍ​ຄືກັນ​ລະຫວ່າງ​ພາສາ Java ແລະ JavaScript ເຊັ່ນ: ຊື່​ຂອງ​ພາສາໄວ​ຍະກອນ ຫຼື​ Library​ ​ມາດຕະຖານ​ຕ່າງ​ໆ ຢ່າງໃດ​ກໍ​ຕາມທັງ​ສອງ​ພາສາ​ແຕກ​ຕ່າງ​ກັນ​ຢ່າງ​ສິ້ນ​ເຊີງ​ໃນ​ແງ່​ຂອງ​ການ​ອອກ​ແບບພາສາ Java ເປັນ​ພາສາ​ທີ່​ມີ​ປະ​ເພດ​ຂໍ້​ມູນ​ແບບ​ຄົງ​ທີ່ (Static-typing) ໃນ​ຂະນະ​ທີ່​ພາສາ JavaScript ມີ​ປະ​ເພດ​ຂໍ້​ມູນ​ແບບ​ໄດ​ນາ​ມິກ​ (Dynamic-typing) ພາສາ Java ຖືກ​ຄອມ​ພາຍເປັນ Byte-code ກ່ອນທີ່ຈະ​ລັນໃນ​ຂະນະ​ທີ່​ພາສາ JavaScript ຈະ​ຄອມ​ພາຍໃນ​ຕອນ​ທີ່​ໂປຣ​ແກຣມ​ຣັນພາສາ Java ເປັນ​ພາສາ​ແບບ Class-based ໃນ​ຂະນະ​ທີ່​ພາສາ JavaScript ເປັນ​ພາສາ​ແບບ Prototypebased.

#### 2.1.6.1 ປະຫວັດ​ຄວາມ​ເປັນ​ມາ​ຂອງ​ພາສາ JavaScript

ພາສາ JavaScript ຖືກ​ອອກ​ແບບ​ແລະ​ສ້າງ​ໂດຍ Brendan Eich ສຳລັບ​ເປັນ​ພາສາ​ສະຄິບ​ທເຮັດວຽກ​ເທິງ​ເວັບ​ບ​ຣາ​ວ​ເຊີ Navigator ທີ່​ເປັນ​ຜະລິດຕະພັນ​ຂອງ​ບໍລິສັດ Netscape ເພື່ອ​ເຮັດໃຫ້​ໜ້າ​ເວັບ​ທີ່​ໃນ​ຕອນ​ທຳອິດ​ນັ້ນ​ເປັນ​ແບບ Static ສາມາດ​ຕອບ​ໂຕ້​ກັບ​ຜູ້​ໃຊ້​ໄດ້​ໂດຍ​ທີ່​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ຣີ​ເຟ​ດ​ໜ້າ​ໃໝ່ (Dynamic) ເຊັ່ນ:ການ​ສົ່ງ​ຂໍ້​ມູນ​ເບື້ອງ​ຫຼັງ​ໄປ​ຍັງ Server ແລະ​ ລໍຖ້າ​ຮັບ​ຜົນ​ຕອບ​ກັບ​ມາ​ດ້ວຍ AJAX; ພາສາ JavaScript ໄດ້​ຖືກ​ເປີດ​ຕົວ​ ແລະ ​ເປັນ​ສ່ວນຫນຶ່ງ​ຂອງ​ເວັບ​ບ​າ​ວ​ເຊີ Navigator ໃນ​ເດືອນ​ກັນຍາ 1995 ໂດຍ​ໃຊ້​ຊື່​ວ່າ LiveScript ແລະ ​ໄດ້​ປ່ຽນ​ເປັນ JavaScript ໃນ​ອີກ​ສາມ​ເດືອນ​ຕໍ່​ມາ.

ໃນ​ເດືອນ​ພ​ະຈິກ 1996 Netscape ໄດ້​ສົ່ງ​ພາສາ JavaScript ໄປ​ຍັງ ECMA International ເພື່ອ​ເປັນ​ຈຸດ​ເລີ່ມ​ຕົ້ນ​ສຳລັບ​ກຳນົດ​ມາດຕະຖານ​ໃຫ້​ທຸກ​ເວັບ​ບຣາວເຊີປະຕິບັດຕາມ​ມາດຕະຖານ​ດັ່ງ​ກ່າວ ເພື່ອ​ໃຫ້​ການ​ພັດທະນາ JavaScript engine ເປັນ​ໄປ​ໃນ​ທິດ​ທາງ​ດຽວ​ກັນນັ້ນ​ໃຫ້​ເກີດ​ການ​ເປີດ​ຕົວ​ຢ່າງ​ເປັນ​ທາງ​ການ​ສຳລັບ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ ECMAScript ໃນ​ເດືອນ​ມິ​ຖຸ​ນາ 1997 ໃນ​ຊ່ວງ​ເວລາ​ຫຼັງ​ຈາກ​ນີ້ ບໍລິສັດ​ຕ່າງ​ໆ ທີ່​ພັດທະນາ​ເວັບ​ບຣາ​ວ​ເຊີຕ່າງ​ກໍ​ຍັງ​ພັດທະນາ JavaScript engine ບໍ່​ເປັນ​ໄປ​ໃນ​ທິດ​ທາງ​ດຽວ​ກັນ​ເທົ່າໃດ ນັ້ນ​ເຮັດໃຫ້​ນັກ​ພັດທະນາ​ເວັບ​ຕ້ອງ​ຂຽນ​ໂຄດຫຼາຍ​ເວີຊັນ​ເພື່ອ​ໃຫ້​ເຮັດວຽກ​ໄດ້​ໃນ​ທຸກ​ເວັບ​ບຣ​າ​ວ​ເຊີຈົນກະທັ້ງ​ໃນ​ເ​ດືອນກໍລະກົດ 2008 ໄດ້​ມີ​ການ​ຈັດການ​ປະ​ຊຸມ​ຂຶ້ນ​ທີ່ Oslo ຈາກ​ອົງ​ກ​ອນ​ ແລະ ​ຝ່າຍ​ຕ່າງ​ໆ ທີ່​ພັດທະນາ JavaScript engine ເຮັດໃຫ້​ເກີດ​ຂໍ້​ຕົກລົງ​ຂຶ້ນ​ໃນ​ຕົ້ນ​ປີ 2009 ເພື່ອ​ລວບລວມ​ງານ​ທີ່​ກ່ຽວ​ຂ້ອງ​ທັງ​ໝົດ​ຂອງ​ພາສາ JavaScript ແລະ ​ຊຸກຍູ້​ພາສາ​ໃຫ້​ຍ່າງ​ໄປ​ຂ້າງ​ໜ້າ ນັ້ນ​ເຮັດໃຫ້​ເກີດ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ ECMAScript ເວີຊັນ​ທີ່ 5 (ES5) ອອກ​ມາ​ໃນ​ເດືອນ​ທັນວາ 2009 ແລະ ​ກ່ອນ​ໜ້າ​ນີ້​ໃນ​ປີ 2008 Google ໄດ້​ເປີດ​ຕົວ​ເວັບ

​ ບ​ຣາ​ວ​ເຊີ Chrome ທີ່​ມາ​ພ້ອມ​ກັບ V8 JavaScript engine ທີ່​ມີ​ແນວ​ຄິດ​ໃນ​ການ​ພັດທະນາ​ແບບ​ຄອມ​ພາຍ​ໃນ​ຕອນ​ທີ່​ໂປຣ​ແກຣມ​ຣັນ (Just-in-time compilation: JIT) ຊຶ່ງ​ມັນ​ເຮັດວຽກ​ໄດ້​ໄວ​ກວ່າ​ ເຮັດໃຫ້​ຜູ້​ພັດທະນາ​ເວບບຣ​າ​ວ​ເຊີ​ອື່ນ​ໆ ຕ້ອງ​ປັບ​ປຸງ JavaScript engine ຂອງ​ພວກ​ເຂົາ​ໃຫ້​ເຮັດວຽກ​ໃນ​ຮູບ​ແບບ JIT ຫຼັງ​ຈາກ​ທີ່​ພັດທະນາ​ຕໍ່​ເນື່ອງ​ມາ​ອີກ​ຫຼາຍ​ປີ ໃນ​ປີ 2015 ໄດ້​ມີ​ການ​ເພີ່ມ​ຄຸນສົມບັດ​ໃໝ່​ໆ ທີ່​ຫຼາກ​ຫຼາຍ​ເຂົ້າ​ມາ ຊຶ່ງ​ຖື​ວ່າ​ເປັນ​ການ​ປ່ຽນ​ແປງ​ເທື່ອ​ສຳຄັນ ແລະ​ ເຮັດໃຫ້​ເກີດ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ ECMAScript 2015 ຫຼື ​ເວີຊັນທີ່ 6 (ES6) ຈົນເຖິງ​ໃນ​ປີ 2015 ຕອນ​ນີ້​ເບິ່ງ​ຄື​ວ່າ​ພາສາ JavaScript ຈະ​ພັດທະນາ​ມາ​ຈົນ​ເຖິງ​ທີ່ສຸດ​ແລ້ວ ເຮັດໃຫ້​ລະຫວ່າງ​ປີ 2016 - 2019 ເວີຊັນ​ໃໝ່​ຂອງ ECMAScript ທີ່​ຖືກ​ເຜີຍແຜ່​ອອກ​ມາ​ໃນ​ແຕ່​ລະ​ປີ​ມີ​ການ​ປ່ຽນ​ແປງແລະ​ເພີ່ມ​ຄຸນສົມບັດ​ພຽງ​ນ້ອຍໆ​ເທົ່າ​ນັ້ນ.

#### 2.1.6.2 ຄຸນສົມບັດ​ຂອງ​ພາສາ JavaScript

ECMAScript 2015 (ES6) ເປັນ​ພາສາ JavaScript ທີ່​ຖື​ວ່າ​ພັດທະນາ​ມາ​ຈົນ​ເຖິງ​ຈຸດ​ສູງ​ສຸດ​ແລ້ວ​ກໍ​ວ່າ​ໄດ້ ມັນ​ຖືກ​ເຜີຍແຜ່​ໃນ​ເດືອນ​ມິ​ຖຸ​ນາ 2015 ຊຶ່ງ​ໃນ​ເວີຊັນ​ນີ້ ໄດ້​ເພີ່ມ​ໄວ​ຍະກອນໃໝ່​ຂອງ​ພາສາ​ຫຼວງ​ຫລາຍເຊັ່ນ: ການສ້າງ​ຄາດ​ດ້ວຍ​ຄຳ​ສັ່ງ class ການສ້າງ​ໂມ​ດູນ ​ແລະ​ ໃຊ້​ງານ​ມັນ​ດ້ວຍ​ຄຳ​ສັ່ງ import ແລະ export ແລະ​ ຄຳ​ສັ່ງ​ສຳລັບ​ປະ​ກາດ​ຕົວ​ປ່ຽນ let ແລະ ​ປະ​ກາດ​ຄ່າ​ຄົງທີ const ຊຶ່ງ​ເຮັດໃຫ້​ຕົວ​ປ່ຽນສາມາດ​ມີ​ຂອບ​ເຂດ​ໃນ​ບ໋ອກ​ທີ່​ມັນ​ຖືກ​ສ້າງ​ຂຶ້ນ​ໄດ້ ແລະ ​ສິ່ງ​ອື່ນ​ໆທີ່​ຖືກ​ເພີ່ມ​ເຂົ້າ​ມາ​ເປັນ​ຈຳນວນ​ຫລາຍເຊັ່ນ: Map, Set, WeakMap, Promise, Reflection, Proxies, Template string ແລະ ​ອື່ນ​ໆ.

ໃນ​ເດືອນ​ມິ​ຖຸ​ນາ 2016 ໄດ້​ມີ​ການ​ເປີດ​ໂຕ​ເວີຊັນ 7 ຫຼື ECMAScript 2016 (ES7) ໄດ້​ມີ​ການ​ເພີ່ມ​ຕົວ​ດຳ​ເນີນ​ການ​ຍົກ​ກຳ​ລັງ (ທີ່​ກ່ອນ​ໜ້າ​ນີ້​ເຮົາ​ຈະ​ໃຊ້​ຜ່ານ​ຟັງຊັນ Math.pow)ຄຳ​ສັ່ງawait async ສຳລັບ​ການ​ຂຽນ​ໂປຣ​ແກຣມ​ທີ່​ເຮັດວຽກ​ບໍ່​ພ້ອມ​ກັນ ແລະ ​ຟັງຊັນ includes ຂອງ​ອາ​ເລ ແລະ ​ໃນ​ປະຈຸບັນພາສາ JavaScript ຖືກ​ພັດທະນາ​ມາ​ຈົນ​ເຖິງ ECMAScript 2020 (ES11) ຊຶ່ງ​ມີ​ການ​ປ່ຽນ​ແປງ​ທີ່​ເພີ່ມ​ຂຶ້ນ​ບໍ່​ເທົ່າໃດ​ຫຼັງ​ຈາກ ES7.

#### 2.1.6.3 JavaScript engine ແມ່ນຫຍັງ?

JavaScript engine ຄື​ໂປຣ​ແກຣມ​ຄອມພິວເຕີ​ທີ່ໃຊ້​ສຳລັບ​ປະ​ມວນ​ຜົນ​ໂຄດຂອງ​ພາສາ JavaScript ຊຶ່ງ JavaScript engine ໃນ​ຊ່ວງ​ເລີ່ມ​ຕົ້ນ​ເປັນພຽງ​ແຄ່​ຕົວປ່ຽນພາສາ (Interpreter) ເທົ່າ​ນັ້ນ ແຕ່​ໃນ​ປະຈຸບັນ​ໄດ້​ມີ​ການ​ພັດທະນາ​ມາ​ໃຫ້​ຢູ່ໃນ​ຮູບ​ແບບ​ຂອງ​ຄອມ​ພາຍເລິ​ທີ່​ມີ​ການ​ຄອມ​ພາຍໃນ​ຕອນ​ທີ່​ໂປຣ​ແກຣມ​ລັນ (Just-in-time compilation: JIT) ເພື່ອ​ເພີ່ມ​ປະ​ສິດ​ທິ​ພາບ​ການ​ເຮັດ​ວຽກງານ​ຂອງ​ໂປຣ​ແກຣມ ໂດຍ​ທົ່ວ​ໄປ​ແລ້ວ JavaScript engine ຈະ​ຖືກ​ພັດທະນາ​ໂດຍ​ຜູ້​ພັດທະນາ​ເວັບບຣາວເຊີທີ່​ປະຕິບັດ​ຕາມ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ​ຂອງ ECMAScript.

### 2.1.7 ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບ Application Programming Interface (API)

ຫຼາຍ​ຄົນ​ອາດ​ຈະ​​ສົງ​ໄສ​ວ່າ API ຄື​ຫຍັງ ຄວາມຈິງແລ້ວຫຍໍ້​ມາ​ຈາກ​​ Application Program Interface (API) ຊຶ່ງ​ຄື ຄຳ​ສັ່ງ (Code) ທີ່​ອະນຸຍາດ​ໃຫ້ software program ສາມາດ​ສື່​ສານ​ລະຫວ່າງ​ກັນ​ໄດ້ ຖ້າ​ຈະ​ເວົ້າ​ໃນ​ພາສາ​ຄົນ​ຂຽນ program ແລ້ວ API ເປັນ​ຊ່ອງ​ທາງ​ສຳລັບ​ຂໍ​ໃຊ້​ບໍລິການ​ຄຳ​ສັ່ງ ຈາກ operation system (OS) ຫຼື application ອື່ນ​ໆ ຊຶ່ງ​ມັນ​ໃຊ້​ງານ​ໂດຍ​ຕິດ​ຕັ້ງ function ແລະ ​ເອີ້ນ​ໃຊ້​ງານ​ຕາມ doncument ທີ່​ຂຽນ​ໄວ້.

**ສ່ວນ​ປະກອບ​ຂອງ APIs**

APIs ສ້າງ​ຂຶ້ນ​ຈາກ​ສ່ວນ​ສຳຄັນ 2 ຢ່າງຄື:

* ຂໍ້​ກຳນົດ​ທີ່​ຈະ​ອະທິບາຍ​ການ​ແລກ​ປ່ຽນ​ຂໍ້​ມູນ​ລະຫວ່າງ program ຊຶ່ງ​ເຮັດ​ອອກ​ມາ​ໃນ​ລັກສະນະ document ເພື່ອ​ບອກ​ວ່າ request/response ຕ້ອງ​ເປັນ​ຢ່າງໃດ.
* Software ທີ່​ຂຽນ​ຂຶ້ນ​ຕາມ​ຂໍ້​ກຳນົດ ແລະ ​ກໍ່ການ​ເຜີຍແຜ່​ອອກ​ໄປ​ໃຫ້​ໃຊ້​ງານ.

ໂດຍ​ປົກກະຕິ​ແລ້ວ Application ທີ່​ມີ APIs ຈະ​ຕ້ອງ​ຖືກ​ຂຽນ​ເປັນ​ພາສາ Programming ແລະ ພັດທະນາ​ເພີ່ມ​ໄດ້​ງ່າຍ ຈຶ່ງ​ຈຳ​ເປັນ​ຕ້ອງ​ມີ່​ການ​ກວດ​ສອບ​ໂຄງ​ສ້າງ API ​ສະ​ນັ້ນ API ທີ່​ດີ ຜູ້​ທີ່​ອອກ​ແບບ​ຕ້ອງ​ໃຫ້​ຄວາມ​ສຳຄັນ​ໃນ​ການ test ເພື່ອ​ກວດ​ສອບ logic ທີ່​ສາມາດ​ເກີດ​ຂຶ້ນ​ໄດ້​ຈາກ​ການ​ໃຊ້​ງານ.

* **ການ​ໃຊ້​ງານ APIs**

ປັດຈຸບັນ​ນີ້ API ຖືກ​ໃຊ້​ງານ​ໃນ application ເພື່ອ​ສື່​ສານ​ກັບ user ໂດຍ​ບໍ່​ຈຳ​ເປ້ນ​ຕ້ອງ​ມີ​ຄວາມ​ຮູ້ ບໍລິສັດ​ໃຫຍ່​ໆ​ຫຼາຍ​ບໍລິສັດ​ມີ​ການ​ເປີດ API ໃຫ້​ພາຍນອກ​ເຂົ້າ​ມາ​ໃຊ້​ງານ ເຊັ່ນ facebook, google, twitter ຜູ້​ພັດທະນາ​ລະບົບ​ທີ່​ສົນ​ໃຈ ສາມາດ​ນຳ​ເອົາ API ເຫຼົ່າ​ນີ້​ໄປ​ໄປ​ຕໍ່ຍອດ ຊຶ່ງ​ທາງ​ບໍລິສັດ​ກໍ​ສາມາດ​ຂະຫຍາຍ​ຖານ​ລູກ​ຄ້າ​ອອກ​ໄປ​ໄດ້​ອີກ ຮູບ​ແບບ​ການ​ນຳ​ເອົາ API ໄປ​ໃຊ້​ງານ​ມີ​ດັ່ງ​ນີ້:

**1. Libraries and frameworks**

API ມັກ​ຈະ​ເອົາ​ໄປ​ໃຊ້​ເປັນ software library ຊຶ່ງ​ຂຽນ​ຂຶ້ນ​ຕາມ document ໃນ​ຮູບ​แບບ​ພາສາ program ທີ່​ຕ່າງ​ກັນ​ອອກ​ໄປ ຕາມ​ຄວາມ​ເໝາະ​ສົມ​ກັບ​ວຽກເພື່ອ​ເອົາ​ໄປ​ເຮັດ​ເປັນ framework ໃຫ້​ກັບ​ລະບົບ​ໃຊ້​ໃນ​ການ​ສື່​ສານ​ຫາ​ກັນ.

**2. Operating Systems**

API ສາມາດ​ໃຊ້​ງານ​ໃນ​ການ​ສື່​ສານ​ລະຫວ່າງ application ແລະ operating system ເຊັ່ນ: POSIX ຫຼື ມາດຕະຖານ​ການ​ສື່ສານ​ຂອງ OS ເອງ​ກໍ​ມີ API ເປັນ command line ເພື່ອ​ຄວບ​ຄຸມ​ການ​ເຮັດ​ວຽກ​ຂອງ OS.

**3. Remote APIs**

Remote APIs ເຮັດ​​ໃຫ້ developer ສາມາດ​ເຂົ້າ​ຄວບ​ຄຸມ​ຊັບພະຍາກອນ​ຜ່ານ​ທາງ protocol ເພື່ອ​ໃຫ້​ມີ​ມາດຕາ​ຖານ​ການ​ສື່ສານ​ດຽວ​ກັນ ເຖິງ​ແມ່ນ​ວ່າ​ຈະ​ເປັນ​ຄົນ​ລະ technology ເຊັ່ນ Database API ສາມາດ​ອະນຸຍາດ​ໃຫ້ developer ເຂົ້າ​ມາ​ດຶງ​ຂໍ້​ມູນ​ໃນ database ຫຼາກ​ຫຼາຍ​ຊະນິດ​ໄດ້ ຜ່ານ function ດຽວ​ກັນ ​ສະ​ນັ້ນ remote API ຈຶ່ງ​ຖືກ​ໃຊ້​ເລື້ອຍ​ໃນ​ maintenance ດ້ວຍ​ການເຮັດວຽກ​ທີ່​ຝັ່ງ client ໃຫ້​ໄປ​ດຶງ​ຂໍ້​ມູນ​ຈາກ server ກັບ​ລົງ​ມາ​ເຮັດວຽກ.

**4. Web APIs**

​ໃນ​ປະຈຸບັນແມ່ນນິຍົມ​ໃຊ້​ກັນ​ຫຼາຍ ເພາະ​ຢູ່ໃນ​ກຸ່ມ​ຂອງ HTTP ແລະ ​ຂະຫຍາຍ​ອອກ​ໄປ​ສູ່​ຮູບ​ແບບ [XML](https://saixiii.com/what-is-xml/) แລະ [JSON](https://saixiii.com/what-is-json/) ຊຶ່ງໂດຍລວມ​ແລ້ວ​ກໍ​ຄື​ຢູ່​ເທິງ web service ເຊັ່ນ:

* SOAP (Simple Object Access Protocol) ໃຊ້ XML format ສົ່ງ​ຂໍ້​ມູນ.
* REST (Representational State Transfer) ສາມາດ​ໃຊ້ XML ຫຼື JSON format ສົ່ງ​ຂໍ້​ມູນ.

**ຕົວ​ຢ່າງ API ທີ່​ນິຍົມ​ໃນ​ປະຈຸບັນ**

* [Google Maps API](https://developers.google.com/maps/): ເປີດ​ໃຫ້​ໃຊ້​ງານ​ເພື່ອ​ນຳ​ເອົາ​ແຜນ​ທີ່​ຂອງ Google ມາ​ລົງ​ໃນ webpage ໂດຍ​ອາໄສ JavaScript ຫຼື Flash.
* [YouTube APIs](https://developers.google.com/youtube/): Google ຍອມ​ໃຫ້ developer ສາມາດ​ນຳ​ເອົາ Clip video ເທິງ YouTube ໄປ​ລົງ​ໃນ website ຫຼື application ໄດ້.
* [Flickr API](http://www.flickr.com/services/api/): ເພື່ອ​ໃຫ້ developer ສາມາດ​ເຂົ້າ​ເຖິງ ຄັງ​ຮູບ​ພາບ​ໃນ community
* [Twitter APIs](https://dev.twitter.com/): ມີ REST API ໃຫ້​ຄົ້ນ​ຫາແລ້ວ​ກວດ​ສອບ​ຂໍ້​ມູນ trends ໄດ້.
* [Amazon Product Advertising API](https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html): ເປີດ API ໃຫ້​ໃຊ້​ຄົ້ນ​ຫາ​ສິນ​ຄ້າ ແລະ ການ​ໂຄສະ​ນາ​ຜ່ານ​ທາງ Wreebsite.

### 2.1.8 ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບ Nodejs

Node.js ເປັນ Open-Source ແລະ Cross-Platform JavaScript Runtime Environment ທີ່​ກຳ​ລັງ​ໄດ້ຮັບ​ຄວາມ​ນິຍົມ​ສູງ ໂດຍ​ທົ່ວ​ໄປ​ເຮົາ​ຈະ​ໃຊ້ JavaScript ໃນ​ຝັ່ງ client ແຕ່ Node.js ເຮັດໃຫ້​ເຮົາ​ໃຊ້ JavaScript ໃນ​ຝັ່ງ Server ໄດ້​ດ້ວຍຊຶ່ງ Node.js ສາມາດ Run​ ໄດ້​ເທິງ Platform ທີ່​ຫຼາກ​ຫຼາຍ​ທັງ Windows, Linux, Unix, Mac OS X ແລະ ອື່ນໆ.



ຮູບທີ 5: ຮູບ Nodejs

ຈຸດ​ເດັ່ນ​ທີ່ສຸດ​ຂອງ Node.js ຄື​ມັນ​ເຮັດວຽກ​ແບບ asynchronous ຜູ້​ອ່ານ​ອາດຈະ​ສົງ​ໃສ​ວ່າ​ແລ້ວ​ມັນ​ດີ​ແນວໃດ? ລອງ​ມາ​ເບິ່ງ​ຕົວ​ຢ່າງ​ການ​ຈັດການ​ໄຟ​ລ​ເມື່ອ​ມີ​ການ​ຮ້ອງ​ຂໍ​ຈາກ client ມາ​ທີ່ server ຂອງ PHP ກັບ Node.js ທຽບ​ກັນ​ເບິ່ງ​ເພື່ອ​ໃຫ້​ເຂົ້າໃຈ​ຫຼາຍ​ຂຶ້ນ.​

* ກໍລະນີ PHP ເມື່ອ​ມີ​ການ​ຮ້ອງ​ຂໍ​ເຂົ້າ​ມາ​ມັນ​ຈະ​ເຮັດ​ດັ່ງ​ນີ້:

1. ສົ່ງ task ໄປ​ທີ່​ລະບົບ​ໄຟ​ຂອງ​ຄອມພິວເຕີ.
2. ລໍຖ້າຈົນກະທັ້ງ​ລະບົບ​ໄຟ​ລ​ເປີດ ​ແລະ ​ອ່ານ​ໄຟ​ລ​ສຳເລັດ.
3. ສົ່ງ​ເນື້ອ​ຫາ​ຂອງ​ໄຟ​​ກັບ​ມາ​ໃຫ້ client.
4. ພ້ອມ​ສຳລັບ​ຮັບ​ຄຳ​ຮ້ອງ​ຂໍ​ຖັດ​ໄປ.

* ກໍລະນີ Node.js ເມື່ອ​ມີ​ການ​ຮ້ອງ​ຂໍ​ເຂົ້າ​ມາ​ມັນ​ຈະ​ເຮັດ​ດັ່ງ​ນີ້:

1. ສົ່ງ task ໄປ​ທີ່​ລະບົບ​ໄຟ​ຂອງ​ຄອມພິວເຕີ.
2. ພ້ອມ​ສຳລັບ​ຮັບ​ຄຳ​ຮ້ອງ​ຂໍ​ຖັດ​ໄປ.
3. ພໍ​ລະບົບ​ໄຟ​ລ​​ເປີດ ​ແລະ ​ອ່ານ​ໄຟ​ລ​ສຳເລັດ server ຈະ​ສົ່ງ​ເນື້ອ​ຫາ​ຂອງ​ໄຟ​ກັບ​ມາ​ໃຫ້ client.

ຈາກ​ຕົວ​ຢ່າງ​ຂ້າງ​ເທິງ​ຈະ​ເຫັນ​ວ່າ Node.js ຈະ​ຕັດ​ຂັ້ນ​ຕອນ​ການ​ລໍຖ້າຖີ້ມ​ແລ້ວ​ໄປ​ເຮັດ​ຄຳ​ຮ້ອງ​ຖັດ​ໄປ​ເລີຍທີ່ ​ເປັນ​ແບບນີ້​ເພາະ Node.js ຈະ​ Run ​ແບບ Single-Threaded ແລະ​ ໃນ​ Libaryມາດຕະຖານ​ກໍ​ຈະ​ມີ​ເຊັດ​ຂອງ Asynchronous I/O Primitives ທີ່​ຊ່ວຍ​ປ້ອງ​ກັນ​ໂຄດ JavaScript ຈາກ​ການ Blocking ເຮັດໃຫ້​ລະບົບ​ຄ່ອງແຄ້ວ ແລະ ​ມີ​ປະ​ສິດ​ທິ​ພາບ​ຫຼາຍ​ຂຶ້ນ.

|  |  |
| --- | --- |
| **Synchronous vs Asynchronous** | |
| Synchronous ຄື​ການ​ run ​ໂຄດ​ຕາມ​ລຳ​ດັບ​ທີ່​ເຮົາ​ຂຽນ​ໄວ້ ເຊັ່ນ  alert(1);  alert(2);  alert(3); ຜົນ​ລັບ​​ທີ່​ໄດ້​ຄື​ໜ້າ​ຈໍ​ຈະ​ສະແດງ​ຜົນ 1 ຈາກ​ນັ້ນ​ຈຶ່ງ​ສະແດງ​ຜົນ 2 ແລ້ວ​ສຸດ​ທ້າຍ​ຈຶ່ງ​ສະແດງ​ຜົນ 3 ຕາມ​ລຳ​ດັບ | Asynchronous ຄື​ການ​ຣັນໂຄ​ດ​ທີ່​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເປັນ​ໄປ​ຕາມ​ລຳ​ດັບ​ທີ່​ເຮົາ​ຂຽນ​ໄວ້ ເຊັ່ນ:  alert(1);  setTimeout(() => alert(2), 0);  alert(3); ໃນ​ຕົວ​ຢ່າງ​ນີ້​ໂຄດ alert(2) ໃຊ້​ເວລາ​ດຳ​ເນີນ​ການ​ດົນ​ກວ່າ ເຮັດໃຫ້​ຜົນ​ລັບ​​ທີ່​ໄດ້​ຄື​ໜ້າ​ຈໍ​ຈະ​ສະແດງ​ຜົນ 1 ຈາກ​ນັ້ນ​ຈຶ່ງ​ສະແດງ​ຜົນ 3 ແລ້ວ​ສຸດ​ທ້າຍ​ຈຶ່ງ​ສະແດງ​ຜົນ 2 |
| **Blocking vs Non-blocking** | |
| Blocking ໝາຍ​ເຖິງ​ການ​ທີ່​ເຮົາ​ບໍ່​ສາມາດ​ດຳ​ເນີນ​ການ​ຕໍ່ໄປ​ໄດ້​ຈົນ​ກວ່າ​ຕົວ​ດຳ​ເນີນ​ການ​ທີ່​ກຳ​ລັງ​ລັນ​ຢູ່​ຈະ​ສຳເລັດ​ເສຍ​ກ່ອນ ເຊັ່ນ  alert(1);  var value = localStorage.getItem('foo');  alert(2); ຄຳ​ສັ່ງ localStorage ຈະ​ເປັນ​ຕົວ blocking ເຮັດໃຫ້​ບໍ່​ສາມາດ​ເຮັດ​ຄຳ​ສັ່ງ alert(2) ໄດ້​ຈົນ​ກວ່າ​ມັນ​ຈະ​ດຳ​ເນີນ​ການ​ສຳເລັດ ດັ່ງ​ນັ້ນ​ຜົນ​ລັບ​​ທີ່​ໄດ້​ຄື​ໜ້າ​ຈໍ​ຈະ​ສະແດງ​ຜົນ 1 ຈາກ​ນັ້ນ​ຈຶ່ງ​ລໍຖ້າຈົນກວ່າ​ຄຳ​ສັ່ງ localStorage ຈະ​ສຳເລັດ​ຈຶ່ງ​ສະແດງ​ຜົນ 2 | Non-blocking ໝາຍ​ເຖິງ​ການ​ໂຕທີ່​ດຳ​ເນີນ​ການ​ສາມາດ​ເຮັດ​ຄຳ​ສັ່ງ​ຖັດ​ໄປ​ໄດ້​ເລີຍ​ໂດຍ​ບໍ່​ຕ້ອງ​ລໍຖ້າ​ໃຫ້​ຄຳ​ສັ່ງ​ເດີມ​ເຮັດ​ສຳເລັດ​ກ່ອນເຊັ່ນ:  alert(1);  fetch('example.com').then(() => alert(2));  alert(3); ໃນ​ຕົວ​ຢ່າງ​ນີ້ ຄຳ​ສັ່ງ fetch ເປັນ non-blocking operation ດັ່ງ​ນັ້ນ​ຜົນ​ລັບ​ທີ່​ໄດ້​ຄື​ໜ້າ​ຈໍ​ຈະ​ສະແດງຜົນ 1 ຈາກ​ນັ້ນ​ຈຶ່ງ​ສະແດງ​ຜົນ 3 ແລ້ວ​ພໍ​ຄຳ​ສັ່ງ fetch ສຳເລັດ​ກໍ​ສະແດງ​ຜົນ 2 |

#### 2.1.8.1 ປະຫວັດ Node.js

ແຕ່​ເດີມ​ພາສາ JavaScript ຖືກ​ພັດທະນາ​ມາ​ເພື່ອ​ໃຊ້​ສຳລັບ Browser ທີ່​ຊື່ Netscape Navigator ໃນ​ປີ 1995 ​ໃນ​ຕອນ​ນັ້ນ Netscape ຕັ້ງ​ໃຈ​ຈະ​ຂາຍ Web Server ທີ່​ມີ Environment ຊື່ Netscape LiveWire ຊຶ່ງ​ສາມາດ​ສ້າງ Dynamic page ໂດຍ​ໃຊ້ JavaScript ທາງ​ຝັ່ງ server ອີກດ້ວຍ ແຕ່​ໜ້າ​ເສຍ​ດາຍ​ທີ່ Netscape LiveWire ບໍ່​ປະ​ສົບ​ຄວາມ​ສຳ​ເລັດ ແລະ ​ການ​ໃຊ້ JavaScript ທາງ​ຝັ່ງ Server ກໍ​ບໍ່​ໄດ້ຮັບ​ຄວາມ​ນິຍົມ​ເລີຍ​ຈົນກະທັ້ງ Node.js ຖືກ​ກຳ​ເນີດ​ຂຶ້ນ​ມາ.

ສິ່ງ​ທີ່​ເຮັດໃຫ້ Node.js ເປັນ​ທີ່​ນິຍົມ​ຂຶ້ນ​ມາ​ຄື​ການ​ທີ່​ມັນ​ມາ​ໃນ​ຊ່ວງ​ເວລາ​ທີ່​ເໝາະ​ສົມ ເມື່ອ​ທຽບ​ກັບ JavaScript ທີ່​ເກີດ​ມາ​ຕັ້ງ​ແຕ່​ປີ 1995 ແລ້ວ Node.js ຫາກໍ່​ເກີດ​ມາ​ເມື່ອ​ປີ 2009 ເທົ່າ​ນັ້ນ ຕ້ອງ​ຂອບໃຈ "Web 2.0" applications ເຊັ່ນ Flickr, Gmail ແລະ ອື່ນໆທີ່​ສະແດງ​ໃຫ້​ໂລກ​ຮູ້​ວ່າ​ເວັບ​ສະ

ໄໝ​ໃໝ່​ຄວນ​ໜ້າ​ຕາ​ເປັນ​ຢ່າງໃດ.

ບໍ່ດົນ​ກ່ອນ​ທີ່ Node.js ຈະ​ເກີດ ນັກ​ພັດທະນາ​ເບື້ອງ​ຫຼັງ browser ຊື່​ດັ່ງ​ທັງ​ຫຼາຍ​ແຂ່ງ​ຂັນ​ກັນ​ເຮັດວຽກ​ຢ່າງ​ໜັກ​ເພື່ອຈະ​ໃຊ້ JavaScript ໃຫ້​ໄດ້​ດີ​ທີ່ສຸດ ​ແລະ ​ຫາ​ທາງ​ເຮັດໃຫ້ JavaScript ສາມາດ​ຣັນ​ໄດ້​ໄວ​ຫຼາຍ​ຂຶ້ນເພື່ອ​ໃຫ້​ຜູ້​ໃຊ້​ງານ​ໄດ້ຮັບ​ປະ​ສິດ​ທິ​ພາບ​ທີ່​ດີ​ທີ່ສຸດ ຊຶ່ງ​ຜົນ​ຈາກ​ການ​ແຂ່ງ​ຂັນ​ນີ້​ເຮັດໃຫ້​ເກີດ​ການ​ພັດທະນາ Chrome V8 (open-source JavaScript engine ຂອງ The Chromium Project) ຂຶ້ນ​ມາ ແລະ Node.js ກໍ​ໃຊ້ engine.

ແຕ່​ການ​ທີ່ Node.js ເປັນ​ທີ່​ນິຍົມ​ຂຶ້ນ​ມາ​ບໍ່​ແມ່ນ​ແຕ່ວ່າ​ມັນ​ມາ​ຖືກ​ທີ່​ຖືກ​ເວລາ​ເທົ່າ​ນັ້ນ​ ແຕ່​ມັນ​ໄດ້​ສະແດງ​ໃຫ້​ເຫັນ​ແລ້ວ​ວ່າ ການ​ອອກ​ແບບ ​ແລະ ​ແນວ​ຄິດ​ຂອງ​ມັນ​ຊ່ວຍ​ນັກ​ພັດທະນາ​ທັງ​ຫຼາຍ​ໃຫ້​ສາມາດ​ໃຊ້ JavaScript ທາງ​ຝັ່ງ server ໄດ້​ງ່າຍ​ຂຶ້ນ​ຫຼາຍ​ອີກດ້ວຍ​.

### 2.1.9 ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບ Reactjs (JavaScript Library)

React ເປັນເທັກໂນໂລຢີໜຶ່ງ ທີ່ມາແຮງຫຼາຍໂດຍສ້າງມາຈາກພື້ນຖານແນວຄວາມຄິດແບບ MVC (Model View Controller) ຊຶ່ງໝາຍຄວາມວ່າ React ມີຫນ້າທີ່ຈັດການກັບ Model ຫຼື View ແຕ່ສ່ວນໃຫຍ່ຈະເປັນ View ກ່ອນໜ້ານັ້ນເວລາຈະຂຽນໜ້າເວັບເຮົາກໍຈະເຮັດຜ່ານ HTML ມີການໃຊ້ CSS ໃນການປັບປ່ຽນໜ້າຕາຂອງ UI ແຕ່​ໃນ React ຖ້າ​ຈະ​ສ້າງ​ໜ້າ​ເວັບ​ຂຶ້ນ​ມາ​ໄດ້​ນັ້ນ ເຮົາ​ຈະ​ໃຊ້​ເປັນ [Component](https://www.mindphp.com/%E0%B8%9A%E0%B8%97%E0%B9%80%E0%B8%A3%E0%B8%B5%E0%B8%A2%E0%B8%99%E0%B8%AD%E0%B8%AD%E0%B8%99%E0%B9%84%E0%B8%A5%E0%B8%99%E0%B9%8C/tutorial-reactjs/4864-component-and-props.html) ປຽບໄດ້​ວ່າ Component ເປັນ Block ສ່ວນ​ຍ່ອຍ​ຂອງ​ເວັບ​ເຮົາ​ທີ່​ຈະ​ສ້າງ​ອອກ​ມາ.

**ສະຫຼຸບ​ງ່າຍໆ​ດັ່ງ​ນີ້:**

* React ເປັນ Javascript Libraly ສ້າງ ​ແລະ​ ພັດທະນາ​ຂຶ້ນ​ໂດຍ Facebook ຂຽນ​ໄດ້​ແຕ່ UI ເທົ່າ​ນັ້ນ ແລະ​ ເປີດ​ໃຫ້​ໃຊ້​ຟ​ຣີ.
* React ໃຊ້​ໂຄດ HTML , CSS ແລະ Javascript.
* React ມີ 3 ຄອນ​ເຊບ​ທີ່​ເຮົາ​ຕ້ອງ​ຮຽນ​ຮູ້ຄື Component State ແລະ Props.
* **ຂໍ້​ດີ​ ແລະ ​ຂໍ້​ເສຍ​ຂອງ React**

**ກ. ຂໍ້​ດີ**

* Component ເຂົ້າໃຈ​ງ່າຍເຮົາ​ສາມາດ​ຮຽນ​ຮູ້​ໄດ້​ດ້ວຍ​ຕົວ​ເອງ.
* Tool ຫຼາຍພຽງ React ຢ່າງ​ດຽວ​ກໍ່ສາມາດຂຽນເວັບ​ໄດ້​ທັງ​ເວັບ​ແລ້ວ ໂດຍ​ບໍ່​ຕ້ອງ​ຫາ Tool ເພີ່ມເຕີມ ແລະ​ ຍັງມີ Tool ພັດທະນາ​ອອກ​ມາ​ຢູ່​ເລື້ອຍໆສາມາດ​ເຮັດ App ໄດ້ React ມີ​ເຄື່ອງ​ມື​

ໜຶ່ງ​ທີ່​ຊື່​ວ່າ React Native ເປັນ​ການ​ຂຽນ JavaScript ແລ້ວ​ແປງ​ເປັນ App ແບບ Native ໄດ້​ທັງ​ເທິງ Android ແລະ Ios.

**ຂ. ຂໍ້​ເສຍ**

* ຕ້ອງ​ມີ​ພື້ນ​ຖານ​ໃນ Javascript ໃນ​ລະ​ດັບ​ໜຶ່ງ ຄົົນທີ່​ສົນ​ໃຈ​ຮຽນ​ກໍ​ຈະ​ລຳ​ບາກ​ໜ້ອຍ​ໜຶ່ງ​ຖ້າ​ບໍ່​ໄດ້​ມີ​ພື້ນ​ຖານ Javascript ອາດຈະ​ຕ້ອງ​ໃຊ້​ເວລາໜ້ອຍໜຶ່ງ.
* Documentation ອ່ານ​ຍາກ React ມີ Documentation ທີ່​ຍັງ​ບໍ່​​ຄ່ອຍ​ດີໃນ​ອະນາຄົດ​ອາດຈະ​ມີ​ການ​ພັດທະນາ​ອີກ.

### 2.1.10 ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບ GraphQL (GraphQL API)

GraphQL ຄືພາສາ​ສຳລັບ​ການ​ເຂົ້າ​ເຖິງ​ຂໍ້​ມູນ (Query Language) ເພື່ອ​ການ​ໃຊ້​ງານ API ຂອງ​ລະບົບ ​ແລະ ​ຈະ​ປະ​ມວນ​ຜົນ​ຄຳ​ສັ່ງ​ທີ່​ຝັ່ງ server ຫຼື ​ທີ່​ເອີ້ນວ່າ server-side runtime ໂດຍ​ໃຊ້​ໂຄງ​ສ້າງ​ຂໍ້​ມູນ​ທີ່​ເຮົາ​ກຳນົດ​ໄວ້. ແຕ່ GraphQL ບໍ່​ໄດ້​ພັດທະນາ​ຂຶ້ນ​ມາ​ເພື່ອ​ແທນ​ທີ່​ພາສາ​ສຳລັບ​ການ​ເຂົ້າ​ເຖິງ​ຂໍ້​ມູນ​ເຊັ່ນ SQL ຫຼື ​ເຮັດ​ຫນ້າທີ່​ເປັນ​ລະບົບ​ຈັດ​ເກັບ​ຖານ​ຂໍ້​ມູນ (storage engine) ແຕ່​ຢ່າງ​ໃດ.

ໃນ​ປະຈຸບັນ​ວິທີ​ທີ່​ເຂົ້າ​ເຖິງ​ຂໍ້​ມູນ​ຜ່ານ​ເວັບ​ເຊີ​ວິດ​ທີ່​ໄດ້ຮັບ​ຄວາມ​ນິຍົມ​ກໍ​ຈະ​ເປັນ REST API ຜ່ານ​ທາງ HTTP Mehtods ຕົວ​ຢ່າງ​ເຊັ່ນ: ຕ້ອງ​ການ​ດຶງ​ຂໍ້​ມູນ​ຜູ້​ໃຊ້​ງານ​ທັງ​ໝົດ ກໍ​ສາມາດ​ຮຽກ​ຜ່ານ API ໄດ້​ດັ່ງ​ນີ້:

GET /users

ຫຼື ຫາກ​ຕ້ອງ​ການ​ດຶງ​ເພື່ອນ​ທັງ​ໝົດ​ຂອງ​ຜູ້​ໃຊ້​ງານ ID 25

GET /users/25/friends

ແຕ່​ຫາກ​​ຕ້ອງ​ການ​ດຶງ​ເບີໂທ​ສັບ​ຂອງ​ເພື່ອນ​ທີ່​ເປັນ​ເພື່ອນ​ກັບ​ຜູ້​ໃຊ້​ງານ ID 25 ຈະ​ຕ້ອງ​ດຶງ ​ແລະ ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບໃດ? ໂດຍ​​ມີ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ດັ່ງ​ນີ້:

ຮູບທີ 6: ຮູບຕົວຢ່າງ GraphQL

ຈາກ​ຕົວ​ຢ່າງ​ນີ້​ຈະ​ເຫັນ​ໄດ້​ວ່າ​ເຮົາ​ຈຳ​ເປັນ​ຕ້ອງ​ມີ API 2 end point ນຳກັນຄື GET /users/:id/friends ເພື່ອ​ດຶງ​ຂໍ້​ມູນ​ເພື່ອນ​ຂອງ​ຜູ້​ໃຊ້ ID 25 ຈາກ​ນັ້ນ​ ກໍ​ວົນ​ລູບ​ດຶງ​ຂໍ້​ມູນ​ຜູ້​ໃຊ້​ໂດຍ​ໃຊ້ GET /users/:id ເພື່ອ​ດຶງ​ຂໍ້​ມູນ​ເບີໂທສັບ​ອີກຄັ້ງ ແລະ ​ວິທີ​ນີ້​​ກໍ​ຈະ​ໄດ້​ຂໍ້​ມູນ​ອື່ນ​ໆ ​ທີ່​ບໍ່​ຈຳ​ເປັນ​ຂອງ​ຜູ້​ໃຊ້​ມາ​ອີກ​ ບໍ່​ວ່າ​ຈະ​ເປັນ​ຊື່, ຮູບ​ພາບ ແລະ ​ອີ​ເມ​ວໃນ​ຄວາມ​ເປັນ​ຈິງ​ແລ້ວ ອົງ​ກອນໃຫຍ່​ໆທີ່​ມີ​ຫຼາກ​ຫຼາຍ​ພະແນກ​ ຕ່າງ​ກໍ​ຮ້ອງ​ຂໍ​ຂໍ້​ມູນ​ທີ່​ບໍ່​ຄືກັນເຊັ່ນ ບາງ​ພະແນກຕ້ອງ​ການ​ສະເພາະ​ຂໍ້​ມູ​ນພະ​ນັກ​ງານ, ບາງ​ພະແນກຕ້ອງ​ການ​ຂໍ້​ມູ​ນພະນັກ​ງານ​ພ້ອມ​ຂໍ້​ມູນເງິນ​ເດືອນ ຫຼື ບາງ​ພະແນກ​ຕ້ອງ​ການ​ຂໍ້​ມູ​ນພະ​ນັກ​ງານ​ກັບ​ຈຳນວນ​ວັນທີ​ຂາດ​ ຫຼື ​ລາ​ເທົ່າ​ນັ້ນເປັນ​ຕົ້ນ ການ​ໃຊ້​ງານ REST API ຈຶ່ງ​ສ້າງ​ຄວາມ​ລຳ​ບາກ​ຕໍ່​ການ​ພັດທະນາ ​ແລະ ​ດູແລຮັກສາ​ເຊີ​ວິດ.

ດ້ວຍ​ເຫດ​ນີ້ GraphQL ຈຶ່ງ​ໄດ້​ເຂົ້າ​ມາ​ເພື່ອ​ແກ້​ບັນຫາ​ໃນ​ຈຸດ​ນີ້​ໂດຍ​ສະເພາະ ເພື່ອ​ໃຫ້​ເຮົາ​ສາ ມາດ​ດຶງ​ຂໍ້​ມູນ​ໄດ້​ກົງ​ກັບ​ຄວາມ​ຕ້ອງ​ການ ຫຼຸດ​ຄວາມ​ຊັບ​ຊ້ອນ​ໃນ​ການ​ຂຽນ​ໂຄດ​ເພື່ອ​ດຶງ​ຂໍ້​ມູນ ສາມາດ​ຈັດການ​ ແລະ ​ດູແລຮັກສາ​ໂຄ້​ດ​ໃນ​ຝັ່ງ Server-Side ໄດ້​ງ່າຍ​ຫຼາຍ​ຍິ່ງ​ຂຶ້ນ ແລະ ​ທີ່​ສຳຄັນ GraphQL ບໍ່​ໄດ້​ຜູກ​ຕິດ​ກັບ Database ແລະ ທີ່​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ໃດ​ໆ​ທັງ​ນັ້ນ.

### 2.1.11 ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບ expressjs (Nodejs Library)

Express.js ເປັນ Web Application Framework ຊື່ດັງທີ່ໄດ້ຮັບຄວາມນິຍົມຫຼາຍສຳລັບເຮັດວຽກເທິງ platform ຂອງ Node.js ຊຶ່ງເປັນ Server ໂຕໜຶ່ງໂດຍທັງ Express.js ແລະ Node.js ຕ່າງກໍໃຊ້ພາສາ JavaScript ໃນການພັດທະນາ ຖ້າເປັນ Web Application Framework ໃນສະໄໝກ່ອນ ຄົົນທີ່ພັດທະນາຈະຕ້ອງມີຄວາມຮູ້ຫຼາຍກວ່າ 1 ພາສາ, ພາສາທີ່ເຮັດວຽກທາງຝັ່ງ Server ຢ່າງ PHP ຫຼື ASP ແລະ ພາສາທີ່ເຮັດວຽກທາງຝັ່ງ Client ຢ່າງ JavaScript เພື່ອຫຼຸດຄວາມຫຍຸ້ງຍາກທັງໝົດເຖິງເວລາໃນການຕ້ອງຮຽນຮູ້ຫຼາຍໆພາສາເຮັດໃຫ້ເກີດ Node.js ກັບ Express.js ພຽງແຕ່ມີຄວາມຮູ້ JavaScript ກໍສາມາດຂຽນໄດ້ທັງ Server ແລະ Client ນອກຈາກນີ້ ຖ້າໃຜເຄີຍຂຽນ JavaScript ຈະຮູ້ວ່າມັນມີການຕອບສະໜອງທີ່ວ່ອງໄວແນ່ນອນວ່າ Express.js ກໍຍົກເອົາມາເປັນຂໍ້ເດັ່ນໃນເລື່ອງຄວາມໄວ ໃນເລື່ອງການຮຽນຮູ້ການຂຽນ Express.js ຈະໃຊ້ຮູບແບບທີ່ງ່າຍໃນການຮຽນຮູ້ຫລາຍທີ່ສຸດສຳລັບການພັດທະນາ Express.js ໃນເວັບໄຊຈະເວົ້າເຖິງການໃຊ້ Routing (ການກຳນົດເສັ້ນທາງຂອງລະບົບ) ແລະ Middleware (ການຮັບສົ່ງຂໍ້ມູນຂອງລະບົບ) ສາມາດຂຽນໄດ້ໃນຮູບແບບ MVC ສ່ວນການເຊື່ອມຕໍ່ກັບຖານຂໍ້ມູນສາມາດໃຊ້ MongoDB ຫຼື ຈະໃຊ້ MySql ກໍໄດ້ສຳລັບນາມສະກຸນຂອງໄຟລຄື .js ຂະນະນີ້ໄດ້ພັດທະນາມາເຖິງເວີຊັ່ນທີ່ 4.x ແລ້ວ.

### ຄວາມຮູ້ພື້ນຖານກ່ຽວກັບ Progressive Web App (PWA)

Progressive Web Apps (PWA) ຄືມາດຕະຖານ​ການເຮັດ​ເວັບ​ຈາກ Google ທີ່​ນຳ​ຈຸດ​ເດັ່ນ​ຂອງ Website ແລະ Application ມາ​ລວມ​ກັນ ຜູ້​ໃຊ້​ສາມາດ​ເຂົ້າ​ເວັບ​ໄຊ​ ​ແລະ ​ໃຊ້​ງານ​​​ເປັນ Application ເລີຍທັງ​ໜ້າ​ຕາ ​ແລະ​ ຟີ​ເຈີ ແຕ່​ໂດຍ​ເບື້ອງ​ຫຼັງ​ແລ້ວ​ຍັງ​ເປັນ​ເທັກ​ໂນ​ໂລ​ຢີ​ຂອງ​ເວັບ​ໄຊ​​ຢູ່ (ໃຊ້ HTML, ລັນ​ດ້ວຍ Web Browser) ຈຸດ​ນີ້​ອາດຈະ​ຍັງ​ບໍ່​ເຫັນ​ພາບ ດຽວ​ຄ່ອຍ​ໆເບິ່ງ​ໄປ​ພ້ອມ​ກັນ.

#### ​2.1.12.1 ທີ່​ມາ​ຂອງ Progressive Web Apps

ມາດຕະຖານ​ນີ້​ຜູ້​ເລີ່ມ​ຕົ້ນ​ຄື Google ນັ້ນ​ເອງ​ ຈະເຫັນວ່າ​ທຸກ​ມື້​ນີ້​ຜູ້​ໃຊ້​ໂທ​ສັບ​ຕິດ​ຕັ້ງ​ແອບ​ໃໝ່​

ໜ້ອຍ​ເຖິງ​ຫຼາຍ​ທີ່ສຸດລວມໄປ​ເຖິງ​ຖ້າ​ເວັບ​ຄອນ​ເທັນ​ເວັບໃດເວັບ​ໜຶ່ງ​ຕ້ອງ​ການ​ທີ່​ຈະ​ເຮັດ Application ເພື່ອ​ໃຫ້​ຜູ້​ຕິດ​ຕາມ​ອ່ານ ກໍ​ເບິ່ງ​ຄ້າຍຄືຈະ​ເປັນ​ການ​ລົງ​ທຶນ​ທີ່​ຫຼາຍ​ເກີນ​ໄປ.

ຈຶ່ງ​ມີ​ແນວ​ຄິດ​ເຮັດ​ແອບ​ທີ່​ຣັນ​ດ້ວຍ Web Browser ຊຶ່ງ​ເບື້ອງ​ຫຼັງ​ຄື​ເວັບ​ໄຊ​ເດີມ​ທີ່​ເຮັດ​ໄວ້​ຢູ່​ແລ້ວ ໂດຍ​ນຳ Logo ຂອງ​ເວັບ​ນັ້ນ​ມາ​ຢູ່​ທີ່​ໜ້າ Homescreen ເມື່ອ​ກົດ​ເຂົ້າໄປ​ຈະ​ພົບ​ກັບ​ເວັບ​ໄຊ​ທີ່​ມີ​ໜ້າ​ຕາ​ແລະ ​ຟີ​ເຈີແບບ​ແອບ​ເລີຍ.

ຄິດ​ພາ​ບວ່າ​ຈາກ​ເດີມ​ທີ່​ເຮົາ​ຈະ​ຕ້ອງ ເຂົ້າ Store > ຄົ້ນ​ຫາ​ແອບ > ຖ້າດາວ​ໂຫຼດ > ຕິດ​ຕັ້ງ > ກົດ​ຮັບ Permission > ເລີ່ມ​ໃຊ້​ງານ ຊຶ່ງ​ຂັ້ນ​ຕອນ​ຈະ​ຫຼຸດລົງ​ເຫຼືອ​ພຽງ​ເຂົ້າ​ເວັບ > ກົດ Add to Homescreen ແລ້ວ​ໃຊ້​ງານ​ໄດ້​ເລີຍ.

#### 2.1.12.2 ຮູ້​ຈັກ​ກັບ Native App ແລະ Mobile Web App

Native App ຄື Mobile Application ທີ່​ສ້າງ​ມາ​ຈາກ​ພາສາ​ທີ່​ອອກ​ແບບ​ມາ​ສຳລັບ Platform ນັ້ນ​ໆ ໂດຍ​ສະເພາະ ສາມາດ​ຂັບປະ​ສິດ​ທິ​ພາບ​ຂອງ​ເຄື່ອງ​ມື​ຕ່າງ​ໆເທິງ Platform ນັ້ນ​ມາ​ໃຊ້​ໄດ້​ຢ່າງ​ເຕັມ​ທີ່ເຊັ່ນ: ພາສາ Swift ເທິງ iOS.

Mobile Web App ຄື Mobile Application ທີ່​ສ້າງ​ມາ​ຈາກ​ພາສາ​ທີ່​ໃຊ້​ເຮັດ​ເວັບ​ໄຊ​ ຂໍ້​ດີ​ຄື​ຂຽນ​ເທື່ອ​ດຽວ​ສາມາດ​ນຳ​ໄປ​ສ້າງ​ແອບ​ເພື່ອ​ຣັນ​ເທິງ iOS ແລະ Android ແຕ່​ອາດຈະ​ມີ​ຂໍ້​ຈຳ​ກັດ​ໃນ​ບາງ​ຟີ​ເຈີ ຊຶ່ງ​ປະຈຸບັນ​ກໍ​ພັດທະນາ​ມາ​ຂ້ອນ​ຂ້າງ​ຫຼາຍ​ແລ້ວ.

ຍ້ອນ​ກັບ​ໄປ​ເມື່ອ​ປະ​ມານ 6-7 ປີ​ກ່ອນ ທີ່​ນັກ​ພັດທະນາ​ເລີ່ມ​ສົນ​ໃຈ​ການ​ຂຽນ​ໂປຣ​ແກຣມ​ເທິງ​ໂທລະສັບ​ Platform ທີ່​ນັກ​ພັດທະນາ​ຕ້ອງ​ຈັບ​ຕາເບິ່ງ​ບໍ່​ໄດ້​ມີ​ແຕ່ iOS ແລະ Android ແຕ່​ຍັງມີ​ພວກ Windows Phone, Black Berry ແລະ ​ອີກ​ຫຼາຍ​ເຈົ້າ​ທີ່​ເລັງ​ຈະ​ມາ​ຕີຕະຫຼາດ ເຮັດໃຫ້​ເກີດ​ບັນຫາ Distribution Platforms ເຮັດ​ໜຶ່ງ​ແອບ ແຕ່​ຕ້ອງ​ລົງ​ທຶນ​ຂຽນ​ຮອງ​ຮັບ​ທຸກ​ເຈົ້າ ແລະ​ ນັກ​ພັດທະນາ​ໜຶ່ງ​ຄົນ​ຈະ​ຕ້ອງ​ຂຽນ​ເປັນ​ຫຼາຍ​ພາສາ.

ຕັ້ງແຕ່​​ນັ້ນ​​ມາຈຶ່ງ​ມີ​ຄອນ​ເຊ​ບ Cross Platform ເກີດ​ຂຶ້ນ ໂດຍ​ການສ້າງ​ໂປຣ​ແກຣມ​ຂຶ້ນ​ມາ​ໜຶ່ງ​ຕົວ​ດ້ວຍ​ພາສາ​ໂປຣ​ແກຣມ​ທີ່​ໃຊ້​ເຮັດ​ເວັບ​ໄຊ​ ແລ້ວ​ສາມາດ​ນຳ​ໂປຣ​ແກຣມ​ໄປ​ໃຊ້​ໄດ້​ກັບ​ທຸກ Platform ຊຶ່ງ​ຕອນ​ນັ້ນ​ກໍ​ມີ​ເຈົ້າ​ດັງ​ໆ ເຊັ່ນ: PhoneGap ຫຼື Ionic ອອກ​ມາ.

ແຕ່​ດ້ວຍ​ຄວາມ​ທີ່​ມັນ​ສ້າງ​ຈາກ​ເທັກ​ໂນ​ໂລ​ຢີ​ເວັບ ​ແລະ​ ຍັງ​ບໍ່​ໄດ້​ເປັນ Native ຂອງ​ພາສາ​ໂປຣ​ແກຣມ​ທີ່​ອອກ​ແບບ​ມາ​ສະເພາະ​ທາງ ຈຶ່ງ​ມີ​ຂໍ້​ຈຳ​ກັດ​ຫຼາຍ​ຢ່າງ​ທີ່​ເຮັດໃຫ້​ພັດທະນາ​ໄດ້​ບໍ່​ສະ​ດວກ​ ຊຶ່ງ​ກໍ​ພັດທະນາ​ກັນ​ມາ​ເລື່ອຍ​ໆຕາມ​ການ​ເວລາ ແລະ​ ເມື່ອ Platform ອື່ນ​ມັນ​ຕາຍ​ໄປ​ຈົນ​ເຫຼືອ​ແຕ່ສອງ​ເຈົ້າ​ກໍ​ງ່າຍ​ຂຶ້ນ​ມາ.

ຈະ​ບອກໄວ້​ວ່າ PWA ນັ້ນ ບໍ່​ແມ່ນ​ທັງ Mobile Web, Native App ຫຼື Web Responsive (ເຖິງ​ແມ່ນ​ຈະ​ຄ້າຍ​ກັນ​ພໍ​ສົມຄວນ) ແຕ່​ເອີ້ນວ່າ​ເປັນ​ມາດຕະຖານ​ໃໝ່​ຂອງ​ການເຮັດ​ເວັບ​ທີ່​ທັງໝົດ​ຈຸດ​ເດັ່ນ​ຂອງ​ທຸກ​ຮູບ​ແບບ​ມາ​ໃຊ້​.

#### 21.12.3 ຟີ​ເຈີຫຼັກ​ໂດຍ​ທັ່ວ​ໄປ​ຂອງ Progressive Web Apps

* ໃຊ້​ງານ​ໄດ້​ທຸກ Browser ແລະ ​ທຸກ Mobile Platform.
* ເຮັດ Cache ເກັບ​ໄວ້​ໃນ​ໂຕນັກ​ພັດທະນາ​ຈະ​ເປັນ​ຄົນ​ກຳນົດ​ວ່າ​ຈະ Cache ສິ່ງ​ໃດ​ໄວ້​ແດ່ເຮັດໃຫ້​ສາມາດ​ໃຊ້​ງານ​ໄດ້​ທັງ Online ແລະ Offline.
* ມີ​ການ​ອັບ​ເດດ​ຂໍ້​ມູນ​ທັນ​ທີ​ເມື່ອ Online.
* Layout ຄື​ກັບ Application ຜູ້​ໃຊ້​ບໍ່​ຕ້ອງ​ຮຽນ​ຮູ້​ໃໝ່.
* ສາມາດ Push Notification ໄດ້ ໂດຍ​ທີ່​ບໍ່​ຕ້ອງ​ເປີດ​ເວັບ​ ຫຼື ແອບ​ຖິ້ມ​ໄວ້.

ໃຜ​ທີ່​ສົນ​ໃຈ​ຕົວ​ຢ່າງ​ດ້ານ​ເທິງ​ສາມາດ​ເຂົ້າໄປ​ຫຼິ້ນ​ໄດ້​ທີ່ [www.pokedex.org](https://www.pokedex.org/) ຫຼື ​ອີກ​ເວັບ​ໄຊ​​ໜຶ່ງ​ທີ່​ໃຫຍ່​​ [www.aliexpress.com](https://www.aliexpress.com/) ລອງ​ກົດ Add to Home Screen ແລ້ວ​ປິດ​ອິນ​ເຕີ​ເນັດ​ເລີຍ (ຕອນ​ນີ້​ສະເພາະ​ເທິງ Android ເທົ່າ​ນັ້ນ).

ລວມເວັບ​ໄຊ​​ອື່ນ​ໆ ທີ່​ເປັນ PWA : [pwa.rocks](https://pwa.rocks/)

ອະນາຄົດ​ຂອງ Progressive Web Apps

ໃນ​​ອະນາຄົດ​ຂອງ PWA ນັ້ນ Google Developers Expert ດ້ານ Web Technologies ທີ່​ໃກ້​ຊິດ​ກັບ​ເທັກ​ໂນ​ໂລ​ຢີ​ໃນ​ຝັ່ງ​ເວັບ​ຂອງ Google ໄດ້​ໃຫ້​ຄວາມ​ເຫັນ​ເອົາ​ໄວ້​ວ່າ:

PWA ​ເປັນ​ສິ່ງ​ທີ່​ສາມາດຮັບໃຊ້ experienceໃຫ້​ຜູ້​ໃຊ້​ໄດ້​ຄື​ກັບ​ແອັບແຕ່​ເຮັດວຽກ​ຢູ່​ເທິງ Browser ຊຶ່ງ​ຈະ​ຫຼຸດ​ຂັ້ນ​ຕອນ​ຄວາມ​ຫຍຸ້ງ​ຍາກ​ເທິງ Store ແລະ ​ໃນ​ສ່ວນ​ຂອງ​ນັກ​ພັດທະນາ​ນັ້ນ​ສາມາດ​ເຮັດ​ໄດ້​ໃນ​ຕົ້ນ​ທຶນ​ທີ່​ຖືກ​ກວ່າ​ເຮັດ​ແອັບ​ຫຼາຍ​ເທົ່າ​.

​ໃນ​ອະນາຄົດ​ນັ້ນ PWA ຈະ​ຖືກ​ນຳ​ມາ​ໃຊ້​ເປັນ​ແອັບ​ນ້ອຍ​ໆ ບໍ່​ໄດ້​ມີ Interactive ຫຼາຍລວມໄປ​ເຖິງ​ເຮັດ​ເປັນ Lite Version ແຍກ​ອອກ​ມາ​ຈາກ​ແອັບ​ໃຫຍ່ ເຊັ່ນ: uber ທີ່​ມີ​ແອັບ​ຫຼັກ​​ຢູ່ ແຕ່​ກໍ​ມີ PWA ແຍກ​ອອກ​ມາທີ່​ຜູ້​ໃຊ້​ສາມາດ​ເຂົ້າ​ເວັບ​ແລ້ວ​ກົດ​ຈອງ​ໄດ້​ຢ່າງ​ວ່ອງໄວບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ຜ່ານ​ຂະ​ບວນ​ການ​ຫຍຸ້ງ​ຍາກ​ໃນ​ການ​ໂຫຼດ​ແອັບ​ເທິງ Store ຫຼື aliexpress.com ທີ່​ສາມາດ​ເບິ່ງ​ສິນ​ຄ້າ​ ແລະ​ ເລືອກ​ຊື້​ໄດ້​ຢ່າງ​ວ່ອງໄວ.

ໃນ​ສ່ວນ​ຂອງ iOS ນັ້ນ​ຂ້ອນ​ຂ້າງ​ເປັນ​ເລື່ອງ​ໃຫຍ່ເພາະ​ຖ້າ​ຝັ່ງ​ນັ້ນ​ບໍ່​ເອົາ​ນຳ​ກໍ​ແຈ້ງ​ເກີດ​ຍາກ ຢ່າງໃດ​ກໍ​ຕາມຕໍ່​ໃຫ້​​ບໍ່​ໄດ້ Support ເວັບ​ທີ່​ເປັນ PWA ກໍ​ສາມາດ​ໃຊ້​ງານ​ເທິງ​ເວັບ​ໄຊ​​ໄດ້​ປົກກະຕິ​ບໍ່​ຄາ​ບັນຫາ​ຫຍັງ.

ຕອນ​ນີ້​ລາຍ​ໃຫຍ່​ຫຼາຍ​ເຈົ້າ​ໃຫ້​ຄວາມ​ສຳຄັນ​ກັບ PWA ກັນ​ຫຼາຍ ແລະ ​ເລີ່ມ​ດຳ​ເນີນ​ການ​ໄປ​ແລ້ວ ເຊັ່ນ: Tinder, Lyft, aliexpress, uber ສຳລັບ Social Network ລາຍ​ໃຫຍ່​ຢ່າງ Facebook ແລະ Twitter ກຳ​ລັງ​ຢູ່ໃນ​ຊ່ວງ​ທົດສອບ ສ່ວນ Instagram ເຮັດ​ອອກ​ມາ​ຮຽບຮ້ອຍ ແລະ ​ຟີ​ເຈີ​​.

### ຄວາມຮູ້ກ່ຽວກັບໂປຣແກຣມ Microsoft Office 2016

Microsoft word ແມ່ນໂປຣແກຣມປະມວນຜົນເຊິ່ງອອກແບບມາເພື່ອຊ່ວຍໃຫ້ສ້າງເອກະສານທີ່ມີຄຸນນະພາບໃນລະດັບມືອາຊີບ ເຄື່ອງມືການຈັດຮູບແແບບເອກະສານທີ່ດີທີ່ສຸດຂອງ word ຈະເຮັດໃຫ້ສາມາດຈັດລະບຽບ ແລະ ຂຽນເອກະສານຂອງທ່ານໄດ້ຢ່າງມີປະສິດທິພາບຫຼາຍຂື້ນ word ຍັງມີເຄື່ອງມືການແກ້ໄຂ ແລະ ກວດສອບຄຳຜິດທີ່ມີປະສິດທິພາບ ສາມາດເຮັດວຽກຮ່ວມກັບຜູ້ອື່ນໄດ້ຢ່າງງ່າຍດາຍ.

#### 2.1.13.1 ປະໂຫຍດຂອງໂປຣແກຣມ Microsoft Word Office 2016

* ມີລະບົບປະຕິບັດຕ່າງໆທີ່ຊ່ວຍໃນການເຮັດວຽກໃຫ້ສະດວກສະບາຍຂື້ນເຊັ່ນ: ການກວດຄຳສະກົດ ການກວດສອບໄວຍະກອນ, ການໃສ່ຂໍ້ຄວາມອັດຕະໂນມັດເປັນຕົ້ນ.
* ສາມາດໃຊ້ສ້າງຈົດໝາຍໄດ້ຢ່າງວ່ອງໄວ ໂດຍສາມາດກຳນົດໃຫ້ຜູ້ວິເສດ( Wizard )ໃນ word ສ້າງແບບຟອມຂອງຈົດໝາຍໄດ້ຫຼາຍຮູບແບບຕາມທີ່ຕ້ອງການ.
* ສາມາດໃຊ້ word ສ້າງຕາຕະລາງທີ່ສະຫຼັບຊັບຊ້ອນໄດ້.
* ປັບປຸງເອກະສານໄດ້ງ່າຍ ແລະ ວ່ອງໄວ ສາມາດຕົກແຕ່ງເອກະສານ ຫຼື ເພື່ອຄວາມສະດວກ.
* ຈະໃຫ້ word ສາມາດແຊກຮູບພາບ, ກຣາບຟິກ ຫຼື ຜັງອົງກອນລົງໃນເອກະສານໄດ້.
* word ປັບແຕ່ງໃຫ້ກໍໄດ້ ໂດຍສາມາດເປັນຜູ້ກຳນົດຮູບແບບຂອງເອກະສານເອງ.
* ຄວາມສາມາດໃນການເຊື່ອມຕໍ່ໂປຣແກຣມອື່ນໆ ໃນຊຸດໂປຣແກມ Microsoft office ສາມາດໂອນຍ້າຍຂໍ້ມູນຕ່າງໆລະຫ່ວາງໂປຣມແກມໄດ້.
* ສ້າງເອກະສານໃຫ້ໃຊ້ວຽກໃນອິນເຕີເນັດໄດ້ຢ່າງງ່າຍດາຍ.

#### 2.1.13.2 ຄວາມສາມາດຂອງ Microsoft word

* ສາມາດພິມ ແລະ ແກ້ໄຂເອກະສານ.
* ສາມາດລຶບ, ຍ້າຍ ແລະ ຮ່າງຂໍ້ຄວາມ.
* ສາມາດພິມຕົວອັກສອນປະເພດຕ່າງໆ.
* ສາມາດຂະຫຍາຍຕົວອັກສອນ.
* ສາມາດຈັດຕົວອັກສອນໃຫ້ເປັນຕົວໜາ.,ຕົວຫງ່ຽງ ແລະ ຂີດກ້ອງໄດ້.
* ສາມາດໃສ່ເຄື່ອງໝາຍ ແລະ ຕົວເລກລຳດັບຕາມຫົວຂໍ້.
* ສາມາດແບ່ງເສົາໄດ້.
* ສາມາກວດການສະກົດ ແລະ ແກ້ໄຂໃຫ້ຖຶກຕ້ອງ.
* ສາມາດຄົ້ນຫາ ແລະ ປ່ຽນແປງຂໍ້ຄວາມທີ່ພິມຜິດ.
* ສາມາດຈັດຂໍ້ຄວາມເອກະສານໃຫ້ພິມໄປທາງຊ້າຍ, ທາງຂວາ ແລະ ລະຫວ່າງກາງ.
* ສາມາດໃສ່ຮູບພາບໃນເອກະສານ.
* ສາມາດຕົກແຕ່ງໂຕອັກສອນ,ພິມຕາຕະລາງ.

### 2.1.14 ຄວາມຮູ້ກ່ຽວກັບໂປຣແກຣມ Microsoft Visio 2016

ໂປຣແກຣມ Microsoft Visio ເປັນເຄື່ອງມືເສີມການເຮັດວຽກຂອງ Microsoft Office ໃນການສ້າງແຜນວາດ (Diagram) ປະເພດຕ່າງໆ, ເຊິ່ງເຮັດໄດ້ງ່າຍດາຍ, ສະດວກ ແລະ ວ່ອງໄວ. ເປັນທີ່ນິຍົມໃຊ້, ມີປະໂຫຍດຫຼາຍສຳລັບການຈັດເອກະສານ, ລວມໄປເຖິງແຜນວາດການອອກແບບ, ຂັ້ນຕອນເຮັດການວິເຄາະ ແລະ ອອກກແບບລະບົບຕ່າງໆ.

ສຳລັບໂປຣແກຣມ Microsoft Visio ມີໃຫ້ເລືອກຢູ່ 2 ປະເພດຄື:

1. Microsoft Visio Standard ເໝາະກັບວຽກດ້ານທຸລະກິດເຊັ່ນ: ຜູ້ບໍລິຫານໂຄງການ, ນັກການຕະຫຼາດ, ພະນັກງານຝ່າຍຊັບພະຍາກອນມະນຸດ ແລະ ທິມງານມີໜ້າທີ່ເບິ່ງແຍງການດຳເນີນງານ ເພື່ອຊ່ວຍໃນການເບີ່ງແຜນວາດ ແລະ ຂ່າວສານ.
2. Microsoft Visio Professional ເໝາະສຳລັບມືອາຊີບທາງດ້ານເຕັກນິກ, ພະນັກງານໄອທີ, ນັກພັດທະນາ ແລະ ວິສະວະກອນທີ່ຊ່ວຍໃນການອອກແບບຂໍ້ມູນລະບົບ ເພື່ອໃຊ້ໃນການເຮັດຕົ້ນແບບ Microsoft Visio Professional ເປັນໂປຣແກຣມ ທີ່ຖືກສ້າງຂື້ນມາເພື່ອຊ່ວຍໃນການສ້າງ Flow chart ຫຼື Diagram ຂອງວຽກໃນສາຂາຕ່າງໆໃຫ້ເຮັດວຽກໄດ້ງ່າຍຂື້ນ. ລັກສະນະຢ່າງ

ໜື່ງໃນການສ້າງ Flow chart ໃນ Microsoft Visio Professional ກໍຄືມີຮູບແບບ Diagram ພື້ນຖານຕ່າງໆ ຈັດກຽມໄວ້ໃຫ້, ເຊິ່ງງ່າຍໃນການອອກແບບ ແລະ ໃຊ້ວຽກ.

ຈຸດດີ Microsoft Visio ຄື: ຊອກຫາຂໍ້ຜິດພາດໄດ້ຢ່າງສະດວກ ແລະ ເຂົ້າໃຈງ່າຍໃນການສະແດງທິດທາງການໄຫຼຂໍ້ມູນ.

### 2.1.15 ຄວາມຮູ້ກ່ຽວກັບໂປຣແກຣມ Visual Studio Code

[Visual Studio Code](https://code.visualstudio.com/) ຫຼື ທີ່​ຫຼາຍ​ຄົນ​ນິຍົມ​ເອີ້ນ​ຫຍໍ້​ວ່າ “[vs code](https://code.visualstudio.com/)” ບອກ​​ກ່ອນ​ວ່າ​ Editor ໂຕ​ນີ້​ມັນ​ອອກ​ມາ​ຕັ້ງ​ແຕ່ 29 ເມ​ສາ ປີ 2015 ແລ້ວ​ພັດທະນາ​ຂຶ້ນ​ໂດຍ​ບໍລິສັດ​ຍັກ​ໃຫຍ່​ໄມ​ໂຄຣ​ຊອບ(Microsoft) ເປັນ​ທັງ​ໂຕ​ແກ້​ໄຂ​ ແລະ ​ປັບ​ແຕ່ງ​ໂຄດ (code optimized editor) ທີ່​ຕັດ​ຄວາມ​ສາມາດ​ມາ​ຈາກ Visual Studio ລຸ້ນ​ປົກກະຕິ (ພວກ GUI designer) ອອກ​ໄປເຫຼືອ​ແຕ່ໂຕ editor ຢ່າງ​ດຽວ ສາມາດ​ເຮັດວຽກ​ໄດ້​ຂ້າມ​ແພລດ​ຟອມທັງ​ໝົດວິນ Windows, Mac ແລະ​ Linux​ ຊັບ​ພອດພາສາ​ຫຼາຍ​ຮ້ອຍ​ພາສາ​ອີກ​ ​ຊຶ່ງ​ທາງ​ໄມ​ໂຄຊອບເອງ​ນັ້ນ​ໄດ້​ເປີດ​ໃຫ້​ໃຊ້​ຟີ​ອີກ​ດ້ວຍ.

ຄວາມ​ສາມາດ​ຂອງ​ “vs code” ນັ້ນ​ຈະ​ມີ​ຄວາມ​ສາມາດ​ໃນ​ການ​ເປີດໄດ້​ຄືກັບ editor ໂຕ​ອື່ນ​ໆ ເຊັ່ນ: sublime, Atom, Notepad++​ທັງໝົດ​ເຖິງ​ຄວາມ​ສາມາດ​ໃນ​ການ​ຕິດ​ຕັ້ງ​ເຄື່ອງ​ມື​ເສີມ (Extension) ໂດຍ​ຮັບຮອງ​ໄດ້​ວ່າ​ມີ​ຊັບພອດຢ່າງແນ່ນອນ ເພາະ​ວ່າ​ມັນ​ຖືກ​ພັດທະນາ​ມາ​ໃຫ້​ຕອບ​ໂຈ​ດນັກ​ພັດທະນາ​ຫຼາຍ​ທີ່ສຸດ ແມ່ນການ​ດີ​ຊ້າຍໜ້າ​ຕາ ໃຫ້ເປັນ​ຮູບ​ແບບ​ທີ່​ເຂົ້າໃຈ ​ແລະ ​ໃຊ້​ງານ​ໄດ້​ງ່າຍ ບໍ່​ຕ້ອງ​ສຶກ​ສາ​ຫຍັງເພີ່ມເຕີມ​ກໍ​ໃຊ້​ງານ​ໄດ້ເລີຍ ສິ່ງ​ທີ່​ເຮັດໃຫ້​ມັນ​ໂດດ​ເດັ່ນ​ກວ່າ​ໂຕ​ອື່ນ​ໆ ຄື​ການ​ທີ່​ອອກ​ແບບ​ໃຫ້​ການ​ຄົ້ນ​ຫາ​ສິ່ງ​ຕ່າງ​ໆ ເຮັດ​ອອກ​ມາ​ໃຫ້​ໃຊ້​ງານ​ໄດ້​ງ່າຍ​ ແລະ ​ເບິ່ງ​ງ່າຍ​ກວ່າ​ໂຕອື່ນ​ໆ ​ການ​ທີ່​ສ້າງ​ໃຫ້​ສາມາດ​ເຊື່ອມ​ຕໍ່​ກັບ Git ໄດ້​ຢ່າງ​ວ່ອງໄວ​ ແລະ ​ງ່າຍ​ດາຍ​ ມີ​ຟັງ​ຊັນໃນ​ການ commit, push & pull ຢູ່ໃນ​ໂຕ ຫຼື ​ຈະ​ເບິ່ງ change ຂອງ​ຟາຍ​ທີ່ເກິດຂື້ນ​ກໍ​ໄດ້​ແບບ​ງ່າຍດາຍບໍ່ຕ້ອງສຶກສາຫັຍງເພີ່ມແຕ່ກໍ່ໃຊ້ງານໄດ້ເລີຍ.



**ຮູບທີ 7: ຮູບໂປຣແກຣມ VS Code**

## 2.2 ບົດຄົ້ນຄວ້າທີ່ກ້ຽວຂ້ອງ

## 2.2.1 Southern Bus Station Booking online System

ປຶ້ມໂຄງການຈົບຊັ້ນລະດັບປະລິນຍາຕີສາຂາວິທະຍາສາດຄອມພິວເຕີ ຫົວຂໍ້ເວບໄຊ Southern Bus Station Booking online System ສົກສຶກສາ 2019 - 2020 ເຊິ່ງມີລາຍລະອຽດດັ່ງຕໍ່ໄປນີ້:

ເຄື່ອງມືທີ່ໃຊ້ໃນການພັດທະນາລະບົບຂາຍປີ້ລົດເມອອນລາຍຂອງສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້ປະກອບດ້ວຍ 2 ພາກສ່ວນໃຫ່ຍຄື: ພາກສ່ວນ Software ແລະ ພາກສ່ວນ Hardware, ເຊິ່ງພາກສ່ວນ Software ນຳໃຊ້ Subline text 3 ສຳລັບອອກແບບຟອມ ພ້ອມທັງຂຽນໂປຣແກຣມໃນການຄວບຄຸມ ແລະ ເຊື່ອມຕໍ່ຖານຂໍ້ມູນ, ສຳລັບສ້າງຖານຂໍ້ມູນແມ່ນນຳໃຊ້ phpMyAdmin ເພື່ອຈັດເກັບຂໍ້ມູນຂອງລະບົບ ແລະ ນຳໃຊ້ Microsoft Visio 2016 ໃນການແຕ້ມແຜນວາດການໄຫຼຂໍ້ມູນ ແລະ ແຕ້ມແຜນວາດຄວາມສຳພັນ ( ER-Diagram ). ພາກສ່ວນ Hardware ແມ່ນນຳໃຊ້ຄອມພິວເຕີ 1 ໜ່ວຍ.

ຜົນທີ່ຈະໄດ້ຮັບຈາກລະບົບຂາຍປີ້ລົດເມອອນລາຍຂອງສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້ຄື: ລະບົບຈະຊ່ວຍຫຼຸດຜ່ອນຄວາມຊໍ້າຊ້ອນ ແລະ ຫຍຸ້ງຍາກຂອງຂໍ້ມູນ ເພື່ອອຳນວຍຄວາມສະດວກໃຫ້ແກ່ລູກຄ້າທີ່ມາໃຊ້ບໍລິການ, ສະດວກວ່ອງໄວແກ່ຜູ້ໃຊ້ງານ ແລະ ສ້າງຄວາມເປັນລະບຽບໃຫ້ແກ່ຂໍ້ມູນ. ນອກຈາກນີ້ຍັງ ບໍລິການສລູກຄ້າ, ຄົ້ນຫາ ແລະ ລາຍງານໄດ້ຢ່າງສະດວກວ່ມອງໄວ ພ້ອມຂໍ້ມູນທີ່ຖືກຕ້ອງຊົດເຈນ.

# **ບົດທີ 3**

# **ວິທີດຳເນີນການຄົ້ນຄວ້າ**

## ວິທີສຶກສາ ແລະ ຄົ້ນຄວ້າ

ເປັນ​ຂະ​ບວນ​ການ​ທີ່​ສະ​ແດງ​ເຖິງ​ການ​ດຳ​ເນີນ​ຂັ້ນ​ຕອນ​ການ​ເຮັດ​ວຽກ​ຂອງ​ລະ​ບົບ​ຕັ້ງ​ແຕ່​ຕົ້ນ​ຈົນ​ຈົບ, ບົດ​ໂຄງ​ການ​ຈົບ​ຊັ້ນ​ນີ້​ພ​ວກ​ຂ້າ​ພະ​ເຈົ້າໄດ້​ນຳ​ໃຊ້​ທິດ​ສະ​ດີ​ການ​ວິ​ເຄາະ ແລະ ອອກ​ແບບ​ລະ​ບົບ​ແບບ​ໂຄງ​ທີ່​ປະ​ກອບ​ມີ 5 ໄລ​ຍະ​ຄື:

ແຜນວາດທີ 1 : ແຜນວາດລວມຂອງລະບົບ

### ໄລຍະທີ 1 ການວາງແຜນໂຄງການ (Project planning phase)

ການວາງແຜນໂຄງການຈັດເປັນຂະບວນການພື້ນຖານໃນຄວາມເຂົ້າໃຈຢ່າງເລິກເຊິ່ງວ່າເປັນຫັຍງຕ້ອງສ້າງລະບົບໃໝ່ ທິມງານຕ້ອງພິຈາລະນາວ່າຈະຕ້ອງດຳເນີນງານຕໍ່ໄປແນວໃດກ່ຽວກັບຂະ ບວນການສ້າງລະບົບໃໝ່ ກ່ອນອື່ນໝົດຕ້ອງມີຈຸດກຳເນີດຂອງລະບົບງານເຊິ່ງໂດຍປົກກະຕິແລ້ວຈຸດກຳເນີດຂອງລະບົບງານມັກເກີດຂຶ້ນຈາກຜູ້ໃຊ້ລະບົບເປັນຜູ້ທີ່ຕິດແທດກັບລະບົບໂດຍກົງເຮັດໃຫ້ມີຄວາມໃກ້ສິດກັບລະບົບວຽກທີ່ດຳເນີນຢູ່ຫຼາຍທີ່ສຸດ ເມື່ອຜູ້ໃຊ້ລະບົບມີຄວາມຕ້ອງການປັບປຸງລະບົບ ງານ ດັ່ງນັ້ນຈຶ່ງຖືເປັນຈຸດເລີ່ມຕົ້ນໃນບົດບາດຂອງນັກວິເຄາະລະບົບວ່າຈະຕ້ອງສຶກສາເຖິງຂອບເຂດ ຂອງບັນຫາທີ່ຜູ້ໃຊ້ລະບົບກຳລັງປະສົບບັນຫາຢູ່ ແລະ ຈະດຳເນີນການແກ້ໄຂແນວໃດ ສຶກສາເຖິງຄວາມເປັນໄປໄດ້ວ່າລະບົບໃໝ່ທີ່ຈະພັດທະນາຂຶ້ນມານັ້ນມີຄວາມເປັນໄປໄດ້ ແລະ ຄຸ້ມຄ່າກັບການຈະລົງທຶນ ຫຼື ບໍ່.

ແນວໃດກໍ່ຕາມໄລຍະຂອງການວາງແຜນໂຄງການປົກກະຕິມັກຈະມີໄລຍະເວລາທີ່ສັ້ນ ແຕ່ກໍ່ຖືວ່າເປັນຂັ້ນຕອນທີ່ສຳຄັນທີ່ຈະໃຫ້ເກີດຜົນສຳເລັດໄດ້ ດັ່ງນັ້ນໃນໄລຍະຂອງການວາງແຜນໂຄງການຈຶ່ງໄດ້ອາໄສນັກວິເຄາະລະບົບທີ່ມີຄວາມຮູ້ ແລະ ປະສົບການສູງ ເນື່ອງຈາກວ່າຫາກນັກວິເຄາະລະບົບບໍ່ເຂົ້າໃຈເຖິງບັນຫາອັນແທ້ຈິງທີ່ເກີດຂຶ້ນ ກໍ່ຈະບໍ່ສາມາດພັດທະນາລະບົບຂຶ້ນມາເພື່ອແກ້ໄຂບັນຫາໃຫ້ຖືກຈຸດໄດ້ ແລະ ມັກຈະມີໂຄງການພັດທະນາລະບົບຫຼາຍໂຄງການທີ່ຫຼັງຈາກໄດ້ດຳເນີນການພັດທະນາ ແລະ ນໍາມາໃຊ້ງານແລ້ວປະກົດວ່າບໍ່ສາມາດຕອບສະໜອງຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ງານເຊິ່ງຖືວ່າເປັນເລື່ອງທີ່ກໍ່ໃຫ້ເກີດຄວາມສູນເສຍທັງທາງດ້ານການລົງທຶນ ແລະ ໄລຍະເວລາ.

ສະຫຼຸບລວມແລ້ວໄລຍະຂອງການວາງແຜນໂຄງການປະກອບມີກິດຈະກໍາຕ່າງໆຕໍ່ໄປນີ້:

- ກໍານົດບັນຫາ Problem Definition.

- ສຶກສາຄວາມເປັນໄປໄດ້ຂອງໂຄງການ Feasibility study.

- ສ້າງຕາຕະລາງກໍານົດເວລາໂຄງການ Project scheduling

- ຈັດຕັ້ງທິມງານໂຄງການ Staff the project

### ໄລຍະທີ 2 ການວິເຄາະ (Analysis Phase)

ໄລຍະການວິເຄາະຈະຕ້ອງມີຄຳຕອບກ່ຽວກັບຄຳຖາມວ່າໃຜເປັນຜູ້ທີ່ໃຊ້ລະບົບ ແລະ ມີຫຍັງແດ່ທີ່ຈະຕ້ອງເຮັດໃນໄລຍະນີ້ນັກວິເຄາະລະບົບຈະຕ້ອງດຳເນີນການໃນຂັ້ນຕອນຂອງການວິເຄາະລະບົບງານປັດຈຸບັນ (Current system) ເພື່ອນໍາມາພັດທະນາແນວຄວາມຄິດສໍາລັບລະບົບໃໝ່ (New System).

ຈຸດປະສົງຫຼັກຂອງການວິເຄາະຄືຈະຕ້ອງສຶກສາ ແລະ ສ້າງຄວາມເຂົ້າໃຈໃນຄວາມຕ້ອງການຕ່າງໆທີ່ໄດ້ລວບລວມມາ ດັ່ງນັ້ນ ການລວບລວມຄວາມຕ້ອງການ (Requirements Gathering) ຈຶ່ງຈັດເປັນວຽກພື້ນຖານຂອງການວິເຄາະລະບົບໂດຍຂໍ້ມູນຄວາມຕ້ອງການເຫຼົ່ານີ້ນັກວິເຄາະລະບົບຈະນຳມາວິເຄາະເພື່ອທີ່ຈະປະເມີນວ່າຄວນມີຫຍັງແດ່ທີ່ລະບົບໃໝ່ຕ້ອງດຳເນີນການ ແລະ ດ້ວຍເຫດນີ້ເອງການກຳນົດລາຍລະອຽດກ່ຽວກັບຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ (User Requirements) ຈະເພີ່ມຄວາມສຳຄັນຫຼາຍຂຶ້ນເປັນລຳດັບສຳລັບລະບົບທີ່ມີຄວາມສັບຊ້ອນສູງ ແລະ ນັກວິເຄາະຕ້ອງເອົາໃນໃສ່ກັບການລວບລວມຄວາມຕ້ອງການຈາກຜູ້ໃຊ້ບໍ່ຄວນກຳນົດຄວາມຕ້ອງການຂຶ້ນເອງໂດຍໃຊ້ຄວາມຄິດສ່ວນຕົວຂອງຕົນເອງເປັນຫຼັກ ຫຼື ປະເມີນຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ລະບົບບໍ່ເຂົ້າກັບຈຸດປະສົງ ແລະ ຫາກມີການພັດທະນາລະບົບຕໍ່ໄປຈົນແລ້ວລະບົບວຽກທີ່ໄດ້ກໍ່ຈະບໍ່ກົງກັບຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ລະບົບຢ່າງແທ້ຈິງເຮັດໃຫ້ຕ້ອງມີຄວາມປັບປຸງ ຫຼື ປ່ຽນແປງພາຍຫຼັງ.

ນັກວິເຄາະລະບົບສາມາດລວບລວມຄວາມຕ້ອງການຕ່າງໆໄດ້ຈາກການສັງເກດ, ການເຮັດວຽກຂອງຜູ້ໃຊ້, ການໃຊ້ເຕັກນິກ, ການສຳພາດ ຫຼື ການໃຊ້ແບບສອບຖາມການອ່ານເອກະສານກ່ຽວກັບການປະຕິບັດງານຂອງລະບົບວຽກ ປັດຈຸບັນລະບຽບກົດເກນຂອງບໍລີສັດ ແລະ ການມອບໝາຍຕໍາ ແໜ່ງໜ້າທີ່ຮັບຜິດຊອບເຊິ່ງໃນຊ່ວງຂອງການເກັບກຳຂໍ້ມູນຄວາມຕ້ອງການກໍ່ຈະພົບກັບຜູ້ໃຊ້ໃນລະດັບຕ່າງໆທີ່ເຮັດໃຫ້ຮູ້ເຖິ່ງບັນຫາ ແລະ ແນວທາງການແກ້ໄຂບັນຫາທີ່ແນະນຳໂດຍຜູ້ໃຊ້ ດັ່ງນັ້ນ ການເກັບກຳຄວາມຕ້ອງການຈຶ່ງເປັນກິດຈະກຳທີ່ສຳຄັນເພື່ອຄົ້ນຫາຄວາມຈິງ ແລະ ຕ້ອງສ້າງຄວາມຕ້ອງການເຊິ່ງກັນ ແລະ ກັນ ເພື່ອສະຫຼຸບອອກມາເປັນຂໍ້ກຳນົດທີ່ມີຄວາມຊັດເຈນໂດຍຂໍ້ກຳນົດເຫຼົ່ານີ້ເມື່ອຜູ້ໃຊ້ທີ່ກ່ຽວຂ້ອງໄດ້ອ່ານແລ້ວຈະຕ້ອງສຶກສາຄວາມໝາຍໄດ້ກົງກັນ.

ຫຼັງຈາກໄດ້ນໍາຄວາມຕ້ອງການຕ່າງໆມາສະຫຼຸບເປັນຂໍ້ກໍານົດທີ່ຊັດເຈນແລ້ວຂັ້ນຕອນຕໍ່ໄປກໍ່ຄືນໍາຂໍ້ກໍານົດເຫຼົ່ານັ້ນໄປພັດທະນາອອກມາເປັນຄວາມຕ້ອງການຂອງລະບົບໃໝ່ ໂດຍເຕັກນິກທີ່ໃຊ້ກໍ່ຄືການພັດທະນາແບບຈໍາລອງຂະບວນການ (Process Model) ເຊິ່ງເປັນແຜ່ ນພາບທີ່ໃຊ້ອະທິບາຍເຖິງຂະບວນການທີ່ຕ້ອງດຳເນີນໃນລະບົບວ່າມີຫຍັງແດ່ ແລະ ຕໍ່ໄປກໍ່ດຳເນີນການພັດທະນາແບບຈຳລອງຂໍ້ມູນ(Data Model) ເພື່ອອະທິບາຍເຖິງຂໍ້ມູນທີ່ຈັດເກັບໄວ້ສໍາລັບສະໜັບສະໜູນການເຮັກວຽກຕ່າງໆ.

ສະຫຼຸບໄລຍະຂອງການວິເຄາະລະບົບປະກອບມີກິດຈະກໍາຕ່າງໆດັ່ງນີ້:

- ວິເຄາະລະບົບງານປັດຈຸບັນ.

- ເກັບກໍາຄວາມຕ້ອງການໃນດ້ານຕ່າງໆ ແລະ ນໍາມາວິເຄາະເພື່ອສະຫຼຸບເປັນຂໍ້ກໍານົດຊັດເຈນ.

- ນຳຂໍ້ກໍານົດມາພັດທະນາອອກມາເປັນຄວາມຕ້ອງການຂອງລະບົບໃໝ່.

- ສ້າງແບບຈຳລອງຂະບວນການຂອງລະບົບໃໝ່ໂດຍການແຕ້ມແຜນພາບກະແສຂໍ້ມູ (Data Flow Diagram: DFD).

- ສ້າງແບບຈໍາລອງຂໍ້ມູນໂດຍການແຕ້ມ Entity Relationship Diagram: ERD.

### ໄລຍະທີ 3 ການອອກແບບ (Design Phase)

ໄລຍະການອອກແບບເປັນການພິຈາລະນາວ່າລະບົບລະດໍາເນີນການໄປໄດ້ແນວໃດ ເຊິ່ງກ່ຽວ ຂ້ອງກັບຍຸດທະວິທີການຂອງການອອກແບບທີ່ວ່າດ້ວຍການຕັດສິນໃຈວ່າຈະພັດທະນາລະບົບໃໝ່ດ້ວຍແນວທາງໃດເຊັ່ນ ພັດທະນາຂຶ້ນເອງ, ຊື້ໂປຣແກຣມສຳເລັດຮູບ ຫຼື ວ່າຈ້າງບໍລິສັດພັດທະນາລະບົບໃຫ້ເປັນຕົ້ນ. ນອກຈາກນີ້ໄລຍະການອອກແບບຈະກ່ຽວຂ້ອງກັບການອອກແບບທາງດ້ານສະຖາປັດຕະຍະ ກຳລະບົບທີ່ກ່ຽວຂ້ອງກັບອຸປະກອນຮາດແວ, ຊອບແວ ແລະ ເຄືອຂ່າຍ.

ການອອກແບບລາຍງານ (Out Design) ການອອກແບບໜ້າຈໍເພື່ອປະຕິສໍາພັນກັບຜູ້ໃຊ້ (User Interface), ການອອກແບບຜັງງານລະບົບ (System Flowchart), ເຊິ່ງລວມເຖິງລາຍລະອຽດຂອງໂປຣແກມ (Specific Program), ຖານຂໍ້ມູນ (Database) ແລະ ຂໍ້ມູນທີ່ກ່ຽວຂ້ອງແນວໃດກໍ່ຕາມເຖິງວ່າກິດຈະກຳບາງສ່ວນຂອງໄລຍະອອກແບບນີ້ ສ່ວນໃຫ່ຍຈະຖືກດຳເນີນການໄປບາງສ່ວນແລ້ວໃນໄລຍະຂອງການວິເຄາະແຕ່ໄລຍະການອອກແບບນີ້ຈະເນັ້ນເຖິງການດຳເນີນການແກ້ໄຂບັນຫາແນວໃດຫຼາຍກ່ວາໂດຍການນຳຜົນຂອງແບບຈຳລອງທາງ Logical Model. ທີ່ໄດ້ຈາກການວິເຄາະມາພັດທະນາມາເປັນແບບຈໍາລອງທາງ Physical Model.

- ການວິເຄາະຈະເນັ້ນແກ້ໄຂບັນຫາຫຍັງແດ່.

- ການອອກແບບຈະເນັ້ນການແກ້ໄຂບັນຫາແນວໃດ.

- ສະຫຼຸບໄລຍະການອອກແບບ.

- ພິຈາລະນາແນວທາງໃນການພັດທະນາລະບົບ.

- ອອກແບບສະຖາປັດຕະຍະກໍາລະບົບ (Architecture Design).

- ອອກແບບຖານຂໍ້ມູນ (Database Design).

- ອອກແບບການສະແດງຜົນ (Output Design).

- ອອກແບບການປ້ອນຂໍ້ມູນ (Input Design).

- ອອກແບບສ່ວນຕິດຕໍໍ່ກັບຜູ້ໃຊ້ (User Interface).

- ສ້າງຕົ້ນແບບ (Prototype).

- ອອກແບບໂປຣແກມ (Structure Chart).

### ໄລຍະທີ 4 ການນຳໄປໃຊ້ (Implementation Phase)

ໃນໄລຍະການນນໍາໄປໃຊ້ຈະເຮັດໃຫ້ລະບົບເກີດຜົນຂຶ້ນມາໂດຍການສ້າງລະບົບທົດສອບລະບົບ ແລະ ການຕິດຕັ້ງລະບົບໂດຍຈຸດປະສົງຫຼັກຂອງກິດຈະກຳໃນໄລຍະນີ້ ບໍ່ແມ່ນພຽງຄວາມໜ້າເຊື່ອຖືຂອງລະບົບ ຫຼື ລະບົບສາມາດເຮັດວຽກໄດ້ດີເທົ່ານັ້ນ ແຕ່ຕ້ອງໝັ້ນໃຈວ່າຜູ້ໃຊ້ລະບົບຕ້ອງໄດ້ຮັບການເຝິກອົບຮົມເພື່ອໃຊ້ງານລະບົບ ແລະ ຄວາມຄາດຫວັງໃນອົງກອນທີ່ຕ້ອງການຜົນຕອບແທນໃນດ້ານດີກັບການໃຊ້ລະບົບໃໝ່ລຳດັບກິດຈະກຳຕ່າງໆ ທຸກກິດຈະກຳຕ້ອງເຂົ້າມາດຳເນີນການຮ່ວມກັນໃນໄລຍະນີ້ເພື່ອໃຫ້ລະບົບການປະຕິບັດງານໄດ້ຮັບຄວາມປະສົບຜົນສຳເລັດໄດ້ໂດຍດີ.

ສະຫຼຸບໄລຍະການນໍາໄປໃຊ້ຈະປະກອບມີກິດຈະກໍາຕ່າງໆດັ່ງຕໍ່ໄປນີ້:

- ສ້າງລະບົບຂຶ້ນມາດ້ວຍການຂຽນໂປຣແກຣມ.

- ກວດສອບຄວາມຖືກຕ້ອງທາງດ້ານ Verification ແລະ Validation ແລະ ດໍາເນີນການທົດສອບລະບົບ.

- ແປງຂໍ້ມູນ (Convert Data).

- ຕິດຕັ້ງລະບົບ (System Installation) ແລະ ສ້າງເອກກະສານຄູ່ມື.

- ຝຶກອົບຮົມຜູ້ໃຊ້ ແລະ ປະເມີນຜົນລະບົບໃໝ່.

ສໍາລັບການສ້າງລະບົບ ຫຼື ການຂຽນໂປຣແກຣມນັ້ນ ສາມາດໃຊ້ວິທີການຂຽນໂປຣແກຣມດ້ວຍພາສາຄອມພິວເຕີເຊັ່ນ: ການໃຊ້ພາສາ Visual Basic, C#, PHP, Java... ນອກຈາກນີ້ ຍັງມີເຕັກນິກອື່ນໆເຊັ່ນ: ເຄື່ອງມືໃນການພັດທະນາ Application ເຊິ່ງເປັນຊອບແວຣ໌ທີ່ເປັນແຫຼ່ງລວມຂອງເຄື່ອງມືຕ່າງໆທີ່ໃຊ້ເພື່ອພັດທະນາ Application ເຮັດໃຫ້ຜູ້ຂຽນໂປຣແກຣມບໍ່ເຮັດວຽກໜັກຄືເມື່ອກ່ອນມີແຕ່ຮຽນຮູ້ ແລະ ປະຍຸກໃຊ້ເຄື່ອງມືເຫຼົ່ານັ້ນກໍ່ສາມາດພັດທະນາລະບົບງ່າຍຂຶ້ນ.

### ໄລຍະທີ 5 ການບໍາລຸງຮັກສາ (Maintenance Phase)

ໂດຍປົກກະຕິແລ້ວໄລຍະການບໍາລຸງຮັກສາຈະບໍ່ນໍາເຂົ້າໄປລວມໃນສ່ວນຂອງ SDLC ຈົນກະ ທັ້ງລະບົບມີການຕິດຕັ້ງເພື່ອໃຊ້ງານແລ້ວເທົ່ານັ້ນ ໄລຍະນີ້ຈະໃຊ້ເວລາຍາວນານທີ່ສຸດເມື່ອທຽບກັບໄລ ຍະອື່ນໆທີ່ຜ່ານມາ ເນື່ອງຈາກລະບົບຈະຕ້ອງໄດ້ຮັບການບຳລຸງຮັກສາຕະຫຼອດໄລຍະເວລາທີ່ມີການໃຊ້ລະບົບສິ່ງທີ່ຄາດຫວັງຂອງໜ່ວຍງານກໍ່ຄືຕ້ອງການໃຫ້ລະບົບໃຊ້ງານຍາວນານຫຼາຍປີລະບົບສາມາດຮອງຮັບເຕັກໂນໂລຊີໃໝ່ໆໃນອະນາຄົດໄດ້ ດັ່ງນັ້ນ ໃນຊ່ວງໄລຍະເວລາດັ່ງກາວຈຶ່ງສາມາດເພີ່ມເຕີ່ມຄວາມສາມາດຂອງລະບົບໃຫ້ມີປະສິດຕິພາບສູງຂຶ້ນພ້ອມທັງການແກ້ໄຂປັບປຸງໂປຣແກຣມໃນກໍລະນີທີ່ເຫັນຂໍ້ຜິດພາດ.

- ການບໍາລຸງຮັກສາລະບົບ (System Maintenannce).

- ການເພີ່ມເຕີມຄວາມສາມາດໃໝ່ໆເຂົ້າໃນລະບົບ (Enhance System).

- ສະໜັບສະໜູນງານຂອງຜູ້ໃຊ້ (Support the User).

ຈາກໄລຍະຕ່າງໆຕາມຂັ້ນຕອນການພັດທະນາລະບົບຕາມແບບແຜນຂອງ SDLC ຈະເຫັນວ່າມີການໃຊ້ຄໍາວ່າໄລຍະ ແລະ ກິດຈະກໍາເຊິ່ງສາມາດອະທິບາຍລາຍລະອຽດເພື່ອໃຫ້ເກີດຄວາມເຂົ້າໃຈກົງກັນດັ່ງນີ້:

- ໄລຍະ (Phase) ຄືກຸ່ມກິດຈະກໍາທີ່ກ່ຽວຂ້ອງກັນ.

- ກິດຈະກໍາ (Activity) ຄືກຸ່ມຂອງງານທີ່ກຽວຂ້ອງກັນ.

- ໜ້າວຽກ (Task) ຄືວຽກທີ່ດໍາເນີນການເຊິ່ງຖືເປັນວຽກທີ່ນ້ອຍທີ່ສຸດ.

## 3.2 ເຄື່ອງມືທີ່ໃຊ້ໃນການພັດທະນາ (Development Tools)

1. **Hardware:**

* ເຄື່ອງຄອມພີວເຕີ 1 ໜ່ວຍລຸ້ນ Lenovo Lenovo Intel(R) Core(TM) I5-3230M
* ເຄື່ອງຄອມພີວເຕີທີ່ໃຊ້ສຳລັບຂຽນໂປຣແກຣມມີ Spec ດັ່ງນີ້:
* CPU @2.60GHz,
* RAM 8GB DDR3L 1600MHz.
* SSD 240 GB, HDD 500 GB.
* Pocket WiFi Modem ເພື່ອໃຊ້ເຊື່ອມຕໍ່ Internet.

1. **Software:**

* ລະບົບປະຕິບັດການ Windows 10 Professional 64 Bit.
* Microsoft Visio 2016 ໃຊ້ແຕ້ມແຜນວາດການໄຫຼຂໍ້ມູນ (DFD, ER, Flowchart).
* Adobe XD ໃຊ້ອອກແບບ UX/UI.
* Studio 3T For MongoDB ແລະ Moon Modeler ໃຊ້ຈັດການຖານຂໍ້ມູນ ແລະ ອອກແບບ Database Model.
* Visual Studio Code ໃຊ້ຂຽນໂຄດດ້ວຍພາສາ JavaScript (ReactJS, NodeJS, GraphQL).
* MS Office 2013 Professional ໃຊ້ເພື່ອສ້າງ.

## 3.3 ການວິເຄາະຂໍ້ມູນ

### 3.3.1 ແຜນວາດລວມຂອງລະບົບ



**ພະນັກງານ**

**ຜູ້ບໍລິຫານ**

**ລູກຄ້າ**

**ຈອງປີ້**

**ຂໍ້ມູນທີ່ຕ້ອງການ**

**ຈັດການຂໍ້ມູນລາຍງານ**

**ຂໍ້ມູນທີ່ຕ້ອງການລາຍ**

**ອອກປີ້**

**ຢືນຢັນການຈອງ**

**ຂໍ້ມູນທີ່ຕ້ອງການລາຍງານ**

ແຜນວາດທີ 2 : ແຜນວາດລວມຂອງລະບົບ

### 3.1.2 ຕາຕະລາງສະແດງລາຍລະອຽດຕ່າງໆທີ່ກ່ຽວກັບລະບົບ

|  |  |  |
| --- | --- | --- |
| External Entity | Process | Data Store |
| ລູກຄ້າ  ພະນັກງານ  ຜູ້ບໍລິຫານ | 1. **ຈັດການຂໍ້ມູນພື້ນຖານ**    1. ຈັດການຂໍ້ມູນພະນັກງານ    2. ຈັດການຂໍ້ມູນບໍລິສັດ    3. ຈັດການຂໍ້ມູນປະເພດລົດ    4. ຈັດການຂໍ້ມູນລົດ    5. ຈັດການຂໍ້ມູນສາຍທາງ 2. **ລົງທະບຽນ**    1. ສະມັກສະມາຊິກ    2. ເຂົ້າສູ່ລະບົບ 3. **ບໍລິການ**    1. ຈອງປີ້    2. ອອກປີ້ 4. **ລາຍງານ**    1. ຂໍ້ມູນການຈອງ    2. ຂໍ້ມູນພະນັກງານ    3. ຂໍ້ມູນສາຍທາງ    4. ຂໍ້ມູນລົດ    5. ຂໍ້ມູນອອກປີ້ | D1 ຂໍ້ມູນພະນັກງານ  D2 ຂໍ້ມູນລູກຄ້າ  D3 ຂໍ້ມູນບໍລິສັດ  D4 ຂໍ້ມູນລົດ  D5 ຂໍ້ມູນປະເພດລົດ  D6 ຂໍ້ມູນບ່ອນນັ່ງ  D7 ຂໍ້ມູນສາຍທາງ  D8 ຂໍ້ມູນເວລາລົດອອກ  D9 ຂໍ້ມູນການຈອງ  D10 ຂໍ້ມູນລາຍລະອຽດການຈອງ  D11 ຂໍ້ມູນອອກປີ້  D12 ຂໍ້ມູນຕິດຕາມຜູ້ໂດຍສານ |

ຕາຕະລາງທີ 4: ສະແດງລາຍລະອຽດຕ່າງໆທີ່ກ່ຽວກັບລະບົບ

### 3.3.3 ແຜນວາດເນື້ອຫາ (Context Diagram)

ແຜນວາດທີ 5: ແຜນວາດເນື້ອຫາ (Context Diagram)

### 3.1.4 ແຜນວາດລຳດັບຊັ້ນໜ້າທີ່ (Functional Hierarchy Diagram)

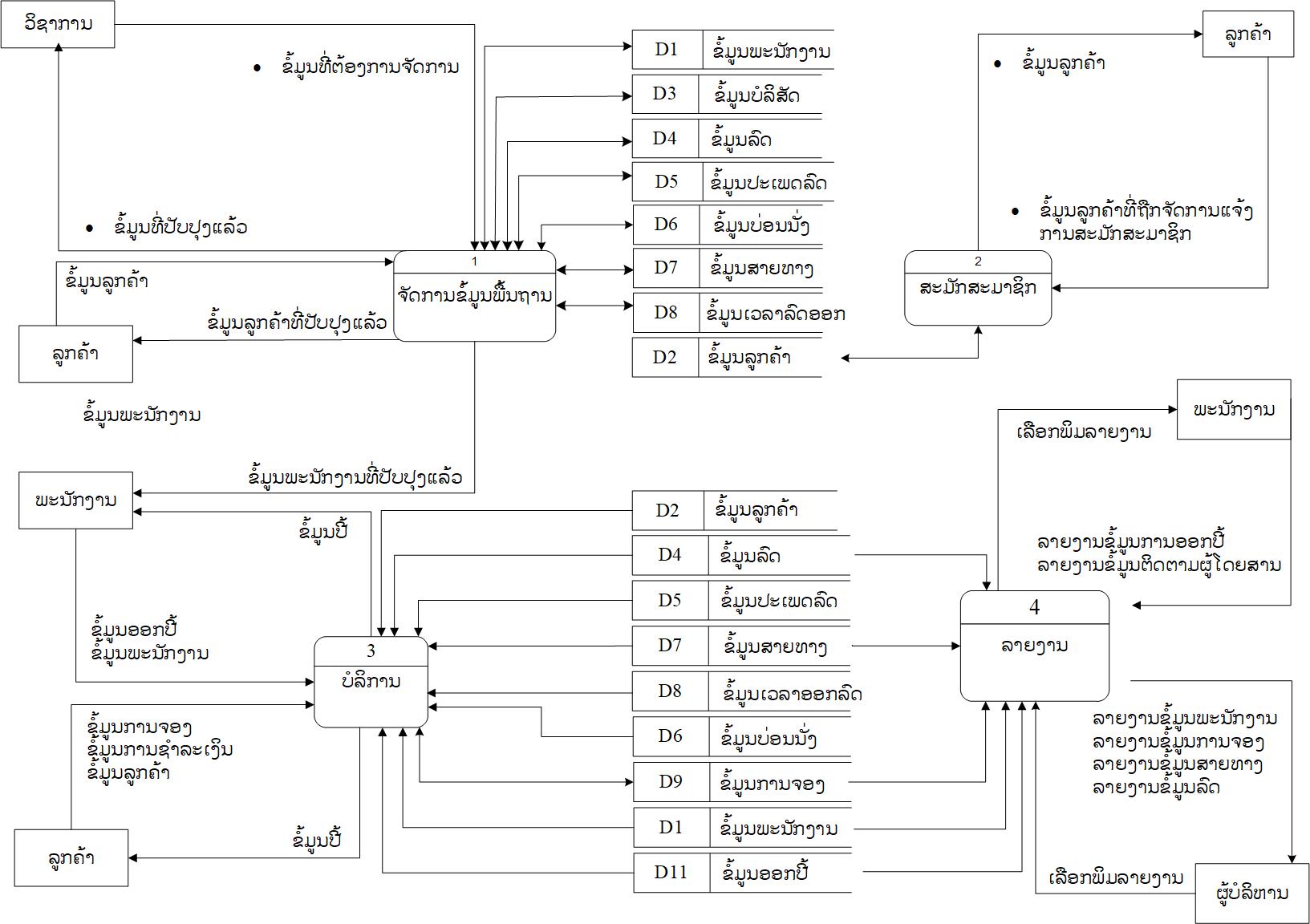
ລະບົບຈອງປີ້ລົດເມສາຍໃຕ້ອອນລາຍ

1. ຈັດການຂໍ້ມູນພື້ນຖານ
   1. ຈັດການຂໍ້ມູນພະນັກງານ
   2. ຈັດການຂໍ້ມູນລົດ
   3. ຈັດການຂໍ້ມູນປະເພດລົດ
   4. ຈັດການຂໍ້ມູນສາຍທາງ
   5. ຈັດການຂໍ້ມູນບໍລິສັດ
   6. ຈັດການຂໍ້ມູນປ່ອງຂາຍປີ້
2. ລົງທະບຽນ
3. ບໍລິການ
   1. ຈອງປີ້
   2. ອອກປີ້
4. ລາຍງານ
   1. ຂໍ້ມູນການຈອງ
   2. ຂໍ້ມູນລົດ
   3. ຂໍ້ມູນພະນັກງານ
   4. ຂໍ້ມູນສາຍທາງ
   5. ຂໍ້ມູນອອກປີ້

ແຜນວາດທີ 6: ແຜນວາດລຳດັບຊັ້ນໜ້າທີ່ (Functional Hierarchy Diagram)

### 3.1.5 ແຜນວາດການໄຫຼຂໍ້ມູນ (Data Flow Diagram: DFD)

#### 3.1.5.1 ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງແຕ່ລະ Process

****

ແຜນວາດທີ 7: ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງແຕ່ລະ Process

#### 3.1.5.2 ແຜນວາດການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງ Process 1



ແຜນວາດທີ 8: ແຜນວາດການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງ Process 1

#### 3.1.5.3 ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງ Process 2

****

ແຜນວາດທີ 9: ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງ Process 2

#### 3.1.5.4 ແຜນວາດການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງ Process 3

ແຜນວາດທີ 10: ແຜນວາດການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງ Process 3

#### 3.1.5.5 ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງ Process 4

****

ແຜນວາດທີ 11: ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 1 ຂອງ Process 4

#### 3.1.5.6 ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 2 ຂອງ Process 1

ແຜນວາດທີ 12: ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 2 ຂອງ Process 1

#### 3.1.5.7 ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 2 ຂອງ Process 3

****

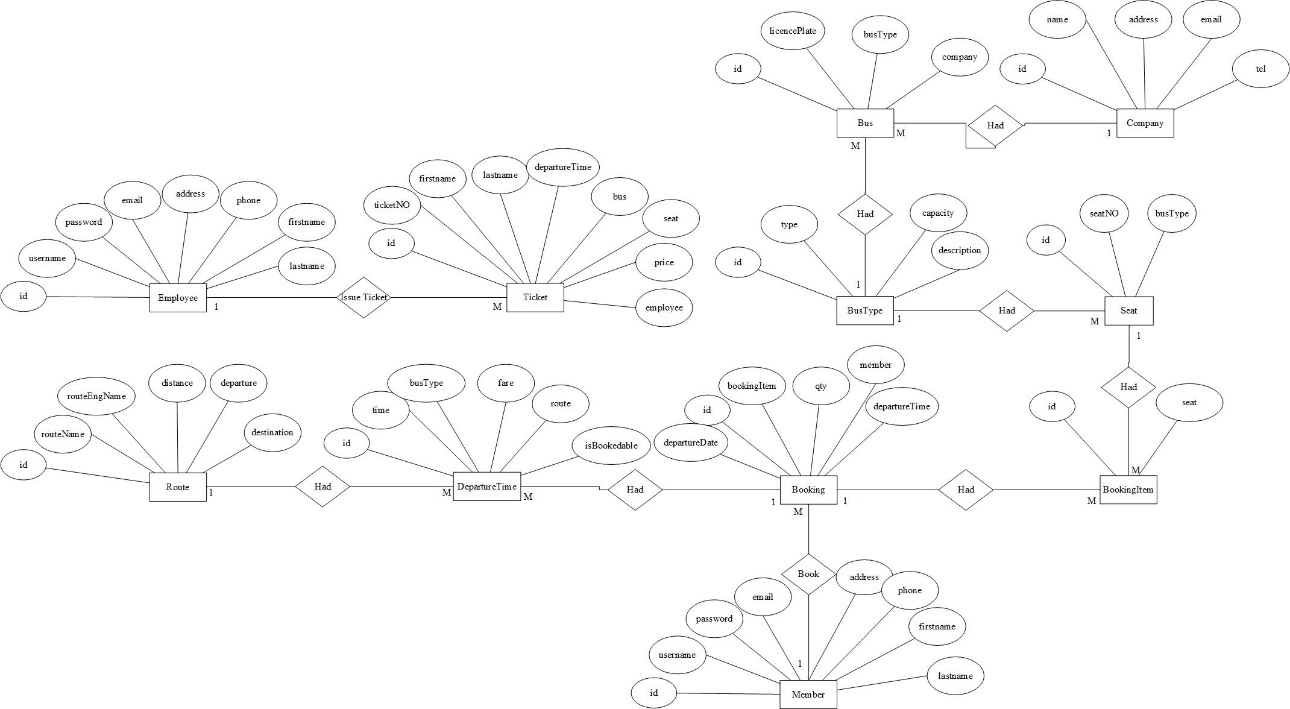
ແຜນວາດທີ 13: ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 2 ຂອງ Process 3

#### 3.1.5.8 ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 2 ຂອງ Process 4

****

ແຜນວາດທີ 14: ແຜນວາດລວມການໄຫຼຂໍ້ມູນລະດັບ 2 ຂອງ Process 4

#### 3.1.5.9 ແຜນວາດຄວາມສຳພັນຂອງຂໍ້ມູນ (ER Diagram)

****

ແຜນວາດທີ 15: ແຜນວາດຄວາມສຳພັນຂອງຂໍ້ມູນ (ER Diagram)

## 3.4 ການອອກແບບລະບົບ

### 3.4.1 ການອອກແບບຮ່າງສະແດງຜົນ (Output Design)

### 3.4.2 ການອອກແບບຮ່າງປ້ອນຂໍ້ມູນ (Input Design)

* **ຟອມເຂົ້າສູ່ລະບົບແອດມິນ**

****

1

2

ຮູບທີ 8 : ຮູບແບບຟອມເຂົ້າສູ່ລະບົບແອດມິນ

* 1. ພາກສ່ວນປ້ອນຊື່ຜູ້ໃຊ້ ແລະ ລະຫັດຜ່ານ
  2. ເຂົ້າສູ່ລະບົບ
* **ຟອມການຈັດການຂໍ້ມູນບໍລິສັດ**

****

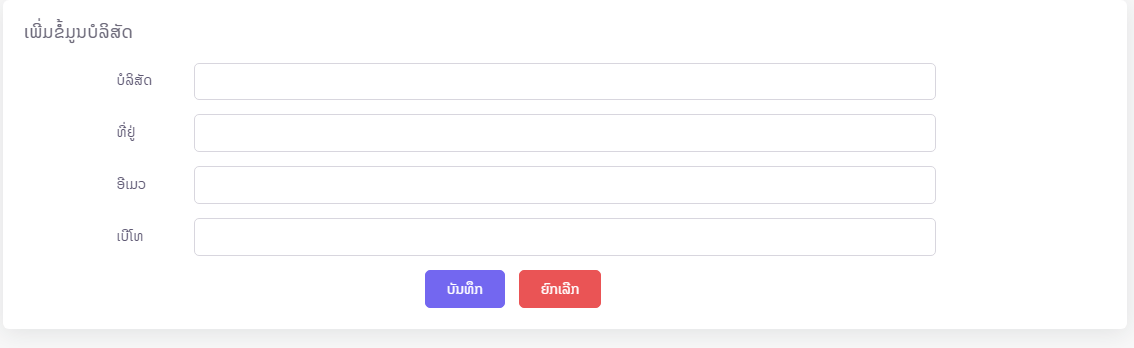
2

1

3

ຮູບທີ 9 : ຮູບແບບຟອມການຈັດການຂໍ້ມູນບໍລິສັດ

* + 1. ປຸ່ມເພີ່ມຂໍ້ມູນ
    2. ປຸມແກ້ໄຂ
    3. ປຸມລຶບ
* **ຟອມການເພີ່ມຂໍ້ມູນບໍລິສັດ**

****

3

2

1

ຮູບທີ 10 : ຮູບແບບຟອມການເພີ່ມຂໍ້ມູນບໍລິສັດ

* **ຟອມການຈັດການຂໍ້ມູນສາຍທາງ**

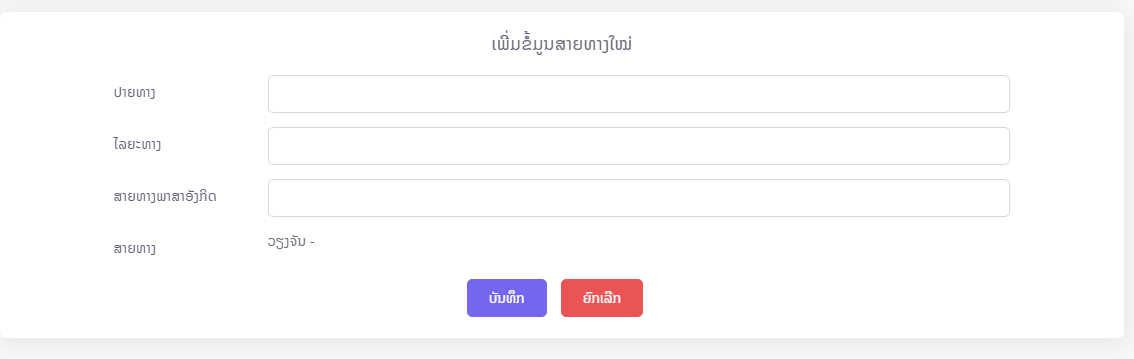
3

4

2

1

ຮູບທີ 11 : ຮູບແບບຟອມການຈັດການຂໍ້ມູນສາຍທາງ

* 1. ພາກສ່ວນປ້ອນຂໍ້ມູນ
  2. ບັນທຶກ
  3. ຍົກເລີກ
  4. ປຸ່ມເພີ່ມຂໍ້ມູນ
  5. ຊ່ອງຄົ້ນຫາ
  6. ການແກ້ໄຂ
  7. ການລຶບ
* **ຟອມການເພີ່ມຂໍ້ມູນສາຍທາງ**

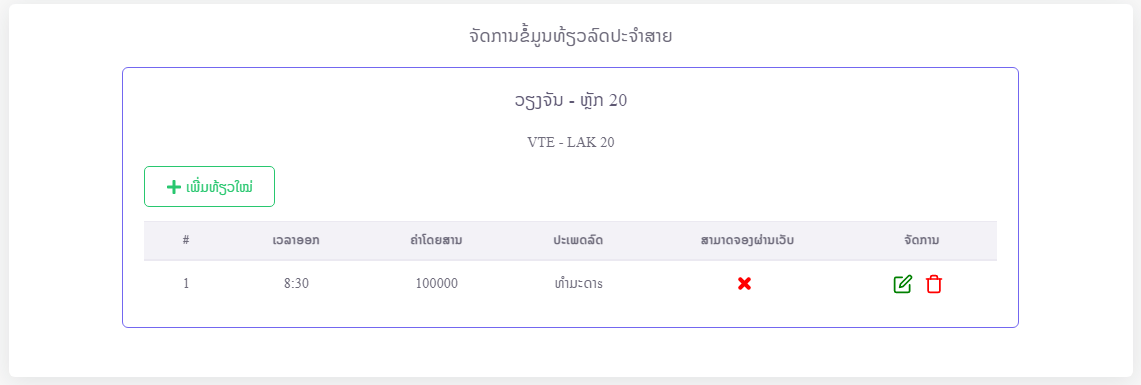
3

2

1

ຮູບທີ 12 : ຮູບແບບຟອມການເພີ່ມຂໍ້ມູນສາຍທາງ

1. ພາກສ່ວນການປ້ອນຂໍ້ມູນ
2. ບັນທຶກ
3. ຍົກເລີກ

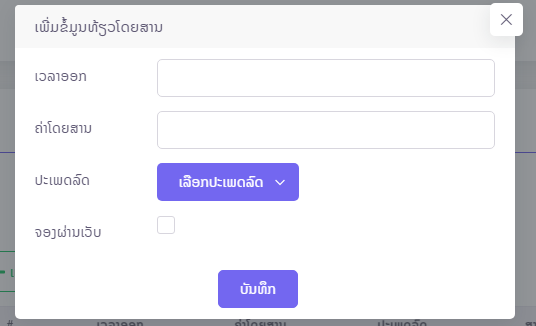
* **ຟອມການຈັດການຖ້ຽວລົດ**

3

2

1

ຮູບທີ 13 : ຮູບແບບຟອມການຈັດການຖ້ຽວລົດ

* 1. ການເພີ່ມຖ້ຽວໃໝ່
  2. ການແກ້ໄຂ
  3. ການລຶບ
* **ຟອມການເພີ່ມຂໍ້ມູນຖ້ຽວລົດ**

1

4

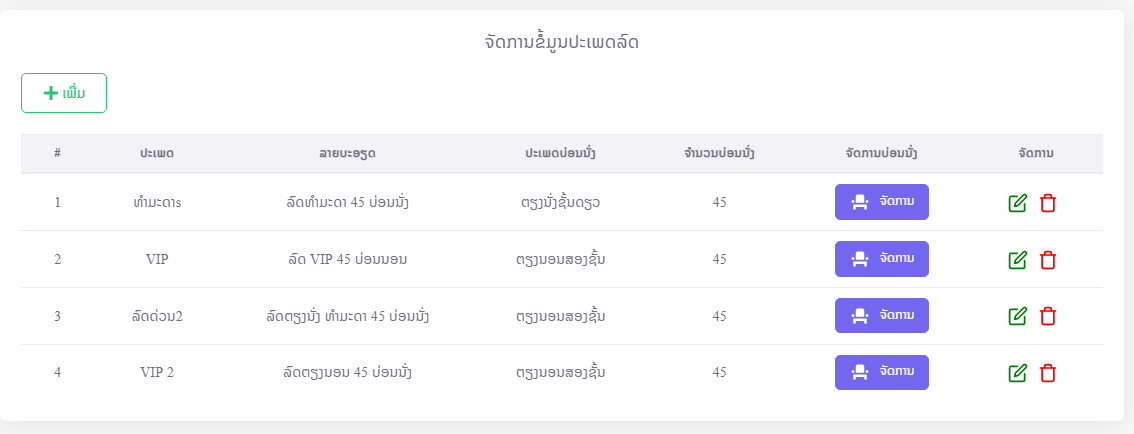
3

2

ຮູບທີ 14 : ຮູບແບບຟອມການເພີ່ມຂໍ້ມູນຖ້ຽວລົດ

1. ພາກສ່ວນປ້ອນຂໍ້ມູນຖ້ຽວໂດຍສານ
2. ເລືອກປະເພດລົດໂດຍສານ
3. ເລຶອກຊ່ອງທາງການສັ່ງຈອງ
4. ບັນທຶກການເພີ່ມຂໍ້ມູນຖ້ຽວລົດ

* **ຟອມການຈັດການປະເພດລົດ**

****

1

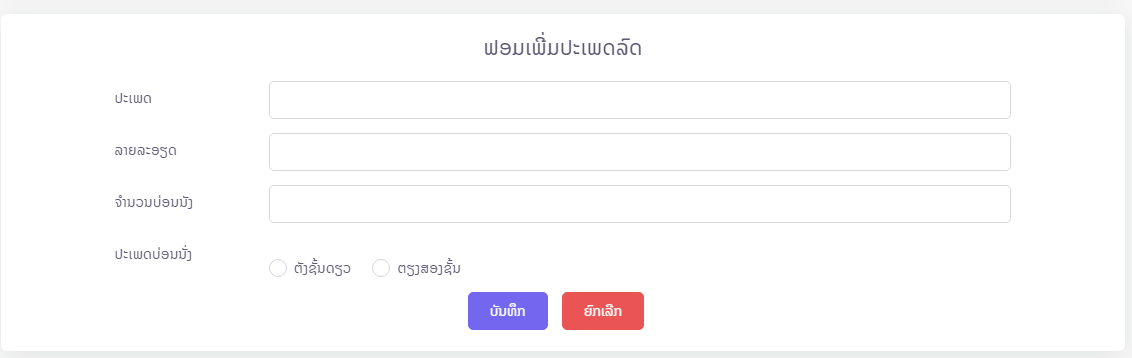
2

3

4

ຮູບທີ 15 : ຮູບແບບຟອມການຈັດການປະເພດລົດ

1. ການເພີ່ມ
2. ການຈັດການປະເພດບ່ອນນັ່ງ
3. ການແກ້ໄຂ
4. ການລຶບ

* **ຟອມການເພີ່ມປະເພດລົດ**

4

3

2

1

ຮູບທີ 16: ຮູບແບບຟອມການເພີ່ມປະເພດລົດ

1. ພາກສ່ວນປ້ອນຂໍ້ມູນການເພີ່ມປະເພດລົດ
2. ເລືອກປະເພດບ່ອນນັ່ງ
3. ບັນທຶກ
4. ຍົກເລີກ

## 3.5 ການອອກແບບຖານຂໍ້ມູນ

#### 3.5.1 ຕາຕະລາງພະນັກງານ (Employee)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Employee** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດພະນັກງານ |  |
| firstname | String | No |  | ຊື່ພະນັກງານ |  |
| lastname | String | No |  | ນາມສະກຸນ |  |
| username | String | No |  | ຊື່ຜູ້ໃຊ້້ພະນັກງານ |  |
| password | String | No |  | ລະຫັດຜ່ານ |  |
| email | String | No |  | ອີເມວ |  |
| address | String | No |  | ທີ່ຢູ່ |  |
| phone | String | No |  | ເບີໂທ |  |
| role | String | No |  | ໜ້າທີ |  |
| status | String | No |  | ສະຖານະ |  |

ຕາຕະລາງທີ 5: ຕາຕະລາງພະນັກງານ (Employee)

#### 3.5.2 ຕາຕະລາງຂໍ້ມູນປີ້ (Ticket)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Ticket** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດປີ້ |  |
| ticketNo | Number | No |  | ເລກທີປີ້ |  |
| firstname | String | No |  | ຊື່ |  |
| lastname | String | No |  | ນາມສະກຸນ |  |
| departureTime | ObjectId | No | FK | ເວລາລົດອອກ | DepartureTime |
| bus | ObjectId | No |  | ລະຫັດລົດ |  |
| Seat | ObjectId | No |  | ລະຫັດບ່ອນນັ່ງ | Seat |
| price | Number | No |  | ລາຄາປີ້ |  |
| employee | ObjectId | No | FK | ລະຫັດຜູ້ອອກປີ້ | Employee |

ຕາຕະລາງທີ 6: ຕາຕະລາງຂໍ້ມູນປີ້ (Ticket)

#### 3.5.3 ຕາຕະລາງຂໍ້ມູນບໍລິສັດ (Company)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Company** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດລູກຄ້າ |  |
| name | String | No |  | ຊື່ |  |
| address | String | No |  | ທີ່ຢູ່ |  |
| tel | String | No |  | ເບີໂທ |  |
| email | String | No |  | ອີເມວ |  |

ຕາຕະລາງທີ 7: ຕາຕະລາງຂໍ້ມູນບໍລິສັດ (Company)

#### 3.5.4 ຕາຕະລາງຂໍ້ມູນປະເພດລົດ (BusType)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Bus Type** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດລູກຄ້າ |  |
| type | String | No |  | ປະເພດ |  |
| description | String | No |  | ລາຍລະອຽດ |  |
| capacity | Number | No |  | ຈຳນວນບ່ອນນັ່ງ |  |
| floorType | Number | No |  | ປະເພດບ່ອນນັ່ງ |  |

ຕາຕະລາງທີ 8: ຕາຕະລາງຂໍ້ມູນປະເພດລົດ (BusType)

#### 3.5.3 ຕາຕະລາງຂໍ້ມູນລົດ (Bus)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Bus** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດລົດ |  |
| licencePlate | String | No |  | ທະບຽນລົດ |  |
| busType | ObjectId | No | FK | ລະຫັດປະເພດລົດ | BusType |
| company | ObjectId | No | FK | ລະຫັດບໍລິສັດ | Company |

ຕາຕະລາງທີ 9: ຕາຕະລາງຂໍ້ມູນລົດ (Bus)

#### 3.5.4 ຕາຕະລາງຂໍ້ມູນບ່ອນນັ່ງ (Seat)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Seat** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດບ່ອນນັ່ງ |  |
| seatNo | String | No |  | ເບີບ່ອນນັ່ງ |  |
| busType | ObjectId | No | FK | ລະຫັດປະເພດລົດ | BusType |
| floor | Number | No |  | ຊັ້ນບ່ອນນັ່ງ |  |

ຕາຕະລາງທີ 10: ຕາຕະລາງຂໍ້ມູນບ່ອນນັ່ງ (Seat)

#### ຕາຕະລາງຂໍ້ມູນສະມາຊິກ (Member)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Member** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດສະມາຊິກ |  |
| firstname | String | No |  | ຊື່ |  |
| lastname | String | No |  | ນາມສະກຸນ |  |
| phone | String | No |  | ເບີໂທ |  |
| username | String | No |  | ຊື່ຜູ້ໃຊ້ |  |
| password | String | No |  | ລະຫັດຜ່ານ |  |
| email | String | No |  | ອີເມວ |  |

ຕາຕະລາງທີ 11: ຕາຕະລາງຂໍ້ມູນສະມາຊິກ (Member)

#### 3.5.6 ຕາຕະລາງຂໍ້ມູນການຈອງ (Booking)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Booking** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດການຈອງ |  |
| bookingItem | ObjectId | No |  | ລາຍລະອຽດການຈອງ |  |
| qty | Number | No |  | ຈຳນວນປີ້ |  |
| member | ObjectId | No | FK | ລະຫັດສະມາຊິກ | Member |
| fullname | String | No |  | ຊື່ເຕັມ |  |
| tel | String | No |  | ເບີໂທ |  |
| email | String | No |  | ອີເມວ |  |
| totalAmount | Number | No |  | ເປັນເງິນ |  |

ຕາຕະລາງທີ 12: ຕາຕະລາງຂໍ້ມູນການຈອງ (Booking)

#### 3.5.6 ຕາຕະລາງຂໍ້ມູນລາຍລະອຽດການຈອງ (BookingItem)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table: BookingItem** | | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດລາຍລະອຽດການຈອງ |  |
| seat [ ] | ObjectId | No | FK | ລະຫັດບ່ອນນັ່ງ | Seat |
| departureTime | ObjectId | No | FK | ເວລາລົດອອກ | DepartureTime |
| departureDate | Date | No |  | ວັນທີເດີນທາງ |  |

ຕາຕະລາງທີ 13: ຕາຕະລາງຂໍ້ມູນລາຍລະອຽດການຈອງ (BookingItem)

#### 3.5.7 ຕາຕະລາງຂໍ້ມູນສາຍທາງ (Route)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table: Route** | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດສາຍທາງ |  |
| routeName | String | No |  | ຊື່ສາຍທາງ |  |
| routeEngName | String | No |  | ຊື່ສາຍທາງອັງກິດ |  |
| distance | Number | No |  | ໄລຍະທາງ |  |
| departure | String | No |  | ຕົ້ນທາງ |  |
| destination | String | No |  | ປາຍທາງ |  |

ຕາຕະລາງທີ 14: ຕາຕະລາງຂໍ້ມູນສາຍທາງ (Route)

#### 3.5.8 ຕາຕະລາງຂໍ້ມູນເວລາລົດອອກ (DepartureTime)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table: DepartureTime** | | | | | | |
| Field Name | Data Type | Allow Null | Key | Description | Reference |
| id | ObjectId | No | PK | ລະຫັດເວລາລົດອອກ |  |
| time | String | No |  | ເວລາ |  |
| busType | ObjectId | No | FK | ລະຫັດປະເພດລົດ | BusType |
| fare | Number | No |  | ຄ່າໂດຍສານ |  |
| route | ObjectId | No | FK | ລະຫັດສາຍທາງ | Route |
| isBookable | Boolean | No |  | ສາມາດຈອງຜ່ານເວັບ |  |

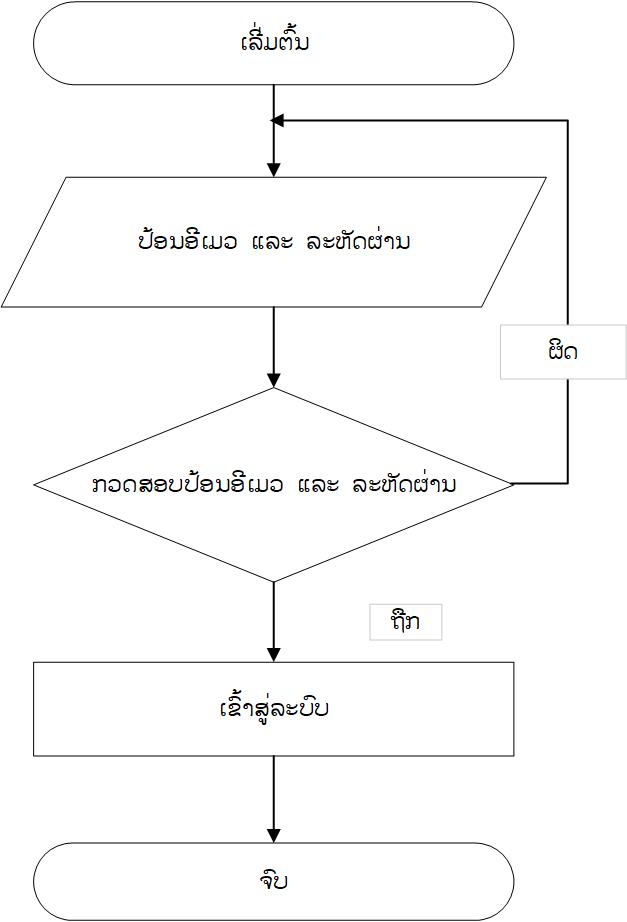
ຕາຕະລາງທີ 14: ຕາຕະລາງຂໍ້ມູນເວລາລົດອອກ (DepartureTime)

## 3.6 ແຜນວາດຂັ້ນຕອນການເຮັດວຽກ (Flowchart)

### 3.6.1 ແຜນວາດ Flowchart ການສະໝັກສະມາຊິກ

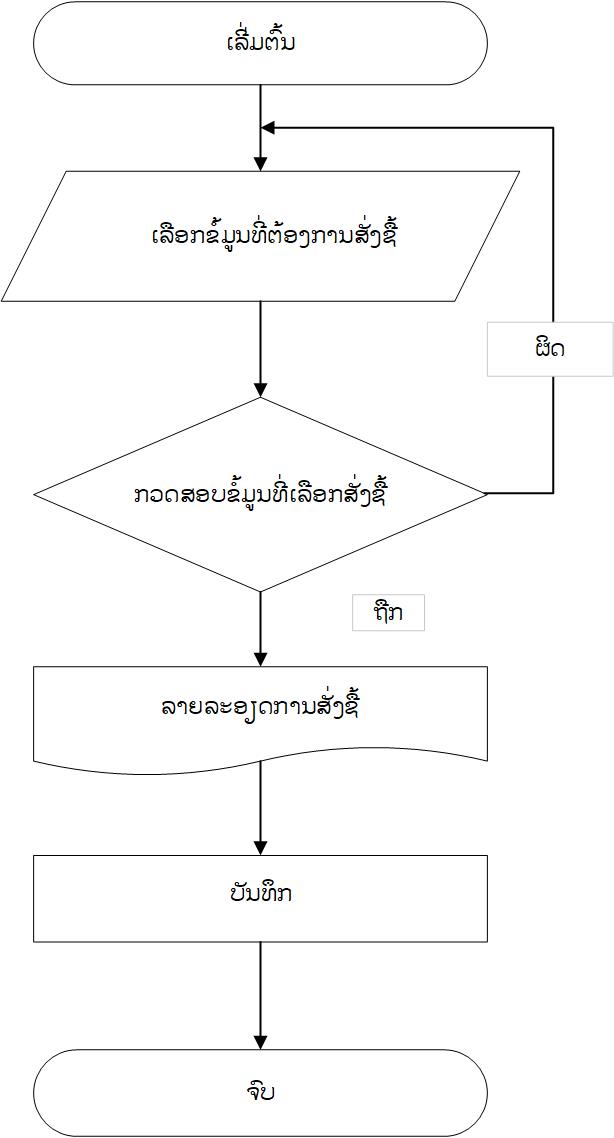
ແຜນວາດທີ 13: ແຜນວາດ Flowchart ການສະໝັກສະມາຊິກ

### 3.6.2 ແຜນວາດ Flowchart ການເຂົ້າສູ່ລະບົບ

****

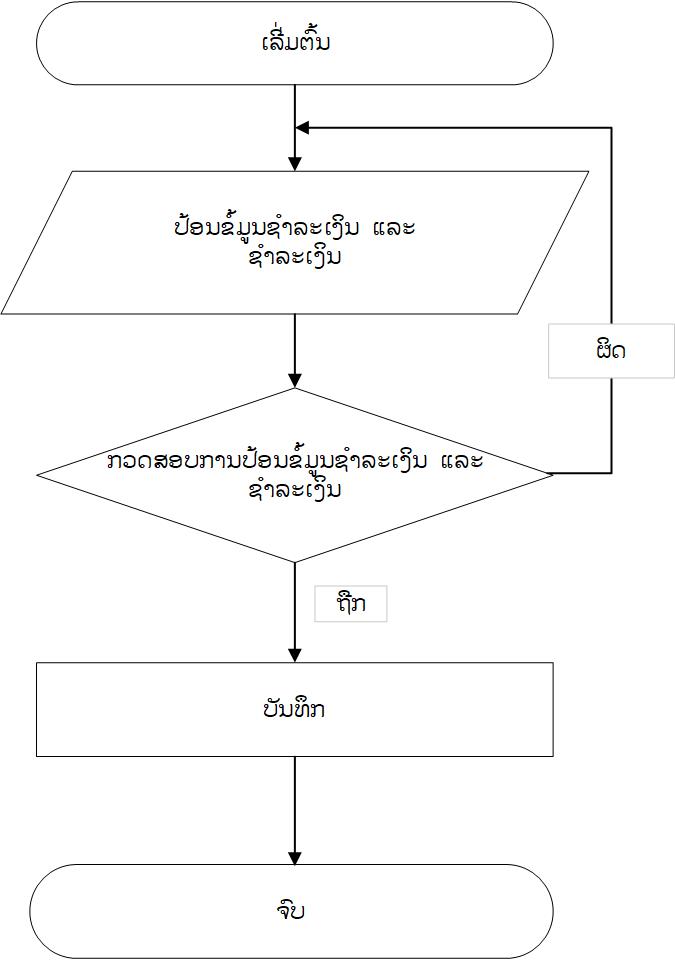
ແຜນວາດທີ 14: ແຜນວາດ Flowchart ການເຂົ້າສູ່ລະບົບ

### 3.6.3 ແຜນວາດ Flowchart ການສັ່ງຊື້

****

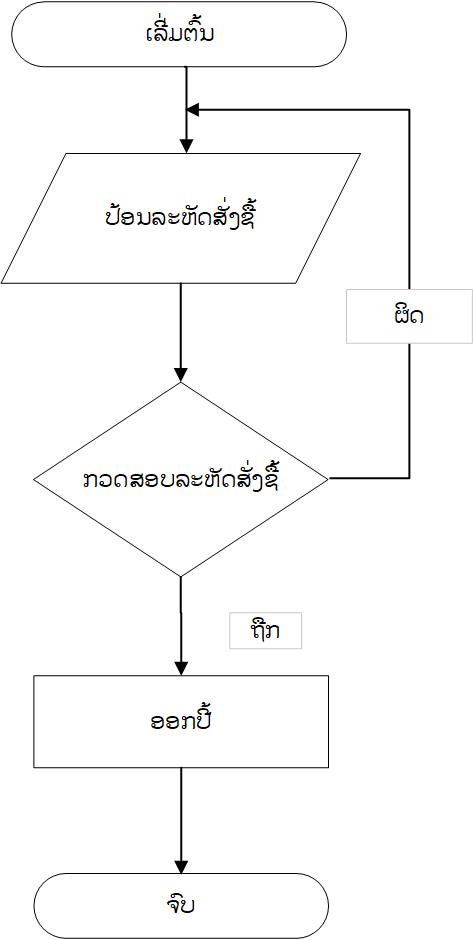
ແຜນວາດທີ 15: ແຜນວາດ Flowchart ການສັ່ງຊື້

### 3.6.4 ແຜນວາດ Flowchart ການຊໍາລະເງິນ

****

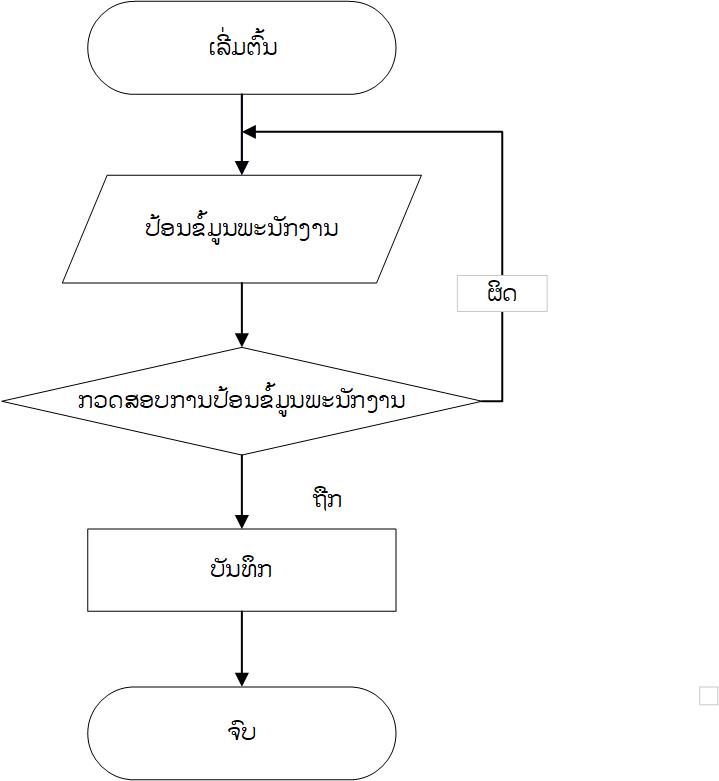
ແຜນວາດທີ 16: ແຜນວາດ Flowchart ການຊໍາລະເງິນ

### 3.6.5 ແຜນວາດ Flowchart ການອອກປີ້



ແຜນວາດທີ 17: ແຜນວາດ Flowchart ການອອກປີ້

### 3.6.6 ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນພະນັກງານ

****

ແຜນວາດທີ 18: ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນພະນັກງານ

### 3.6.7 ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນລູກຄ້າ

ແຜນວາດທີ 19: ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນລູກຄ້າ

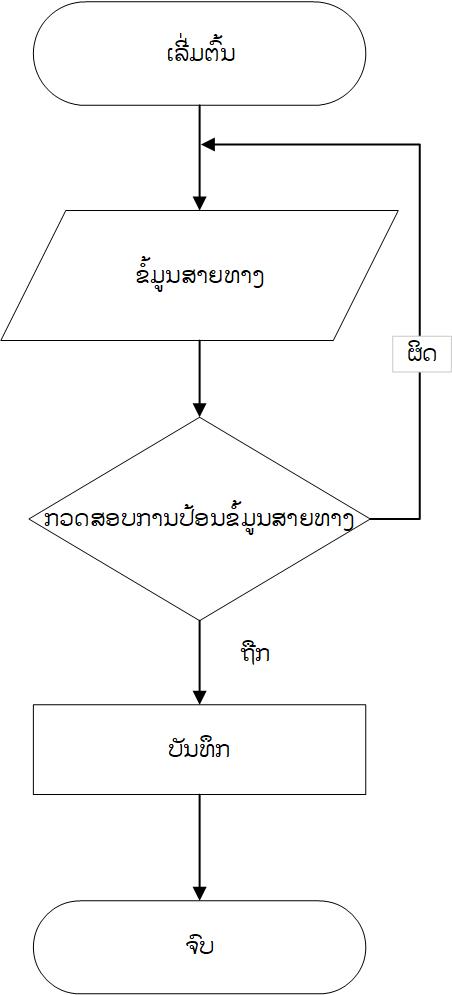
### 3.6.8 ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນລົດ

ແຜນວາດທີ 20: ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນລົດ

### 3.6.9 ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນປະເພດລົດ

ແຜນວາດທີ 21: ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນປະເພດລົດ

### 3.5.10 ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນສາຍທາງ

****

ແຜນວາດທີ 22: ແຜນວາດ Flowchart ຈັດການຂໍ້ມູນສາຍທາງ

# **ບົດທີ 4**

# **ຜົນຂອງການວິເຄາະຂໍ້ມູນ ແລະ ການອະທິບາຍຜົນ**

## 4.1 ໜ້າຕ່າງການ (Login Form)

ເມື່ອເປີດໂປຣແກຣມຂຶ້ນມາກໍ່ຈະເຫັນວ່າເຂົ້າສູ່ລະບົບເປັນໜ້າທຳອິດຈາກນັ້ນຈຶ່ງທຳການເຂົ້າສູ່ລະບົບດ້ວຍຊື່ ແລະ ລະຫັດເຂົ້າໃຊ້ເພື່ອເຂົ້າໄປດ້ານໃນຂອງໂປຣແກຣມດັ່ງຮູບລຸ່ມນີ້:

- ວິທີການເຂົ້າສູ່ລະບົບມີຄື :

1) ໃສ່ຊື່ຜູ້ໃຊ້ລະບົບ.

2) ໃສ່ລະຫັດຜ່ານ.

3) ກົດປຸ່ມເພື່ອເຂົ້າລະບົບ.

4) ຖ້າຊື່ກັບລະຫັດຖືກຈະເປີດນຳໃຊ້ໂປຣແກຣມໄດ້ປົກກະຕິແລ້ວ.

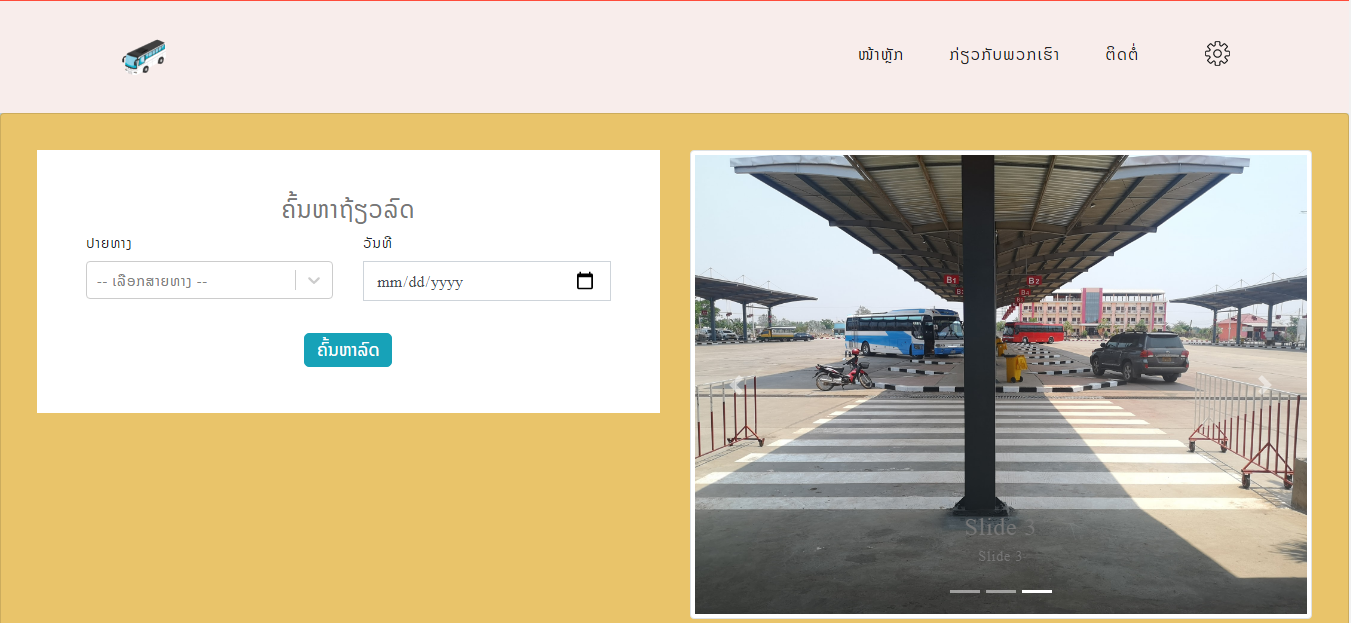
 5) ຖ້າຊື່ກັບລະຫັດຜິດຈະມີຂໍ້ຄວາມຂື້ນມາວ່າ ກະລຸນາກວດສອບຂໍ້ມູນຂອງທ່ານໃຫ້ຖືກຕ້ອງ ແລ້ວກົດປຸ່ມ OK.

ຮູບທີ 8: ຮູບໜ້າຕ່າງການ (Login Form)

## 4.2 ຟອມຫຼັກ (Main Form)

ຟອມຫຼັກແມ່ນຟອມທີ່ເຮົາສາມາດເຂົ້າຫາທຸກໆຟອມໄດ້ເຊັ່ນ: ຟອມຈັດການຂໍ້ມູນ, ຟອມປ້ອນຂໍ້ມູນ, ຟອມການບໍລິການ, ແລະ ຟອມລາຍງານ.

### 4.2.1 ຟອມຄົ້ນຫາຖ້ຽວລົດ



3

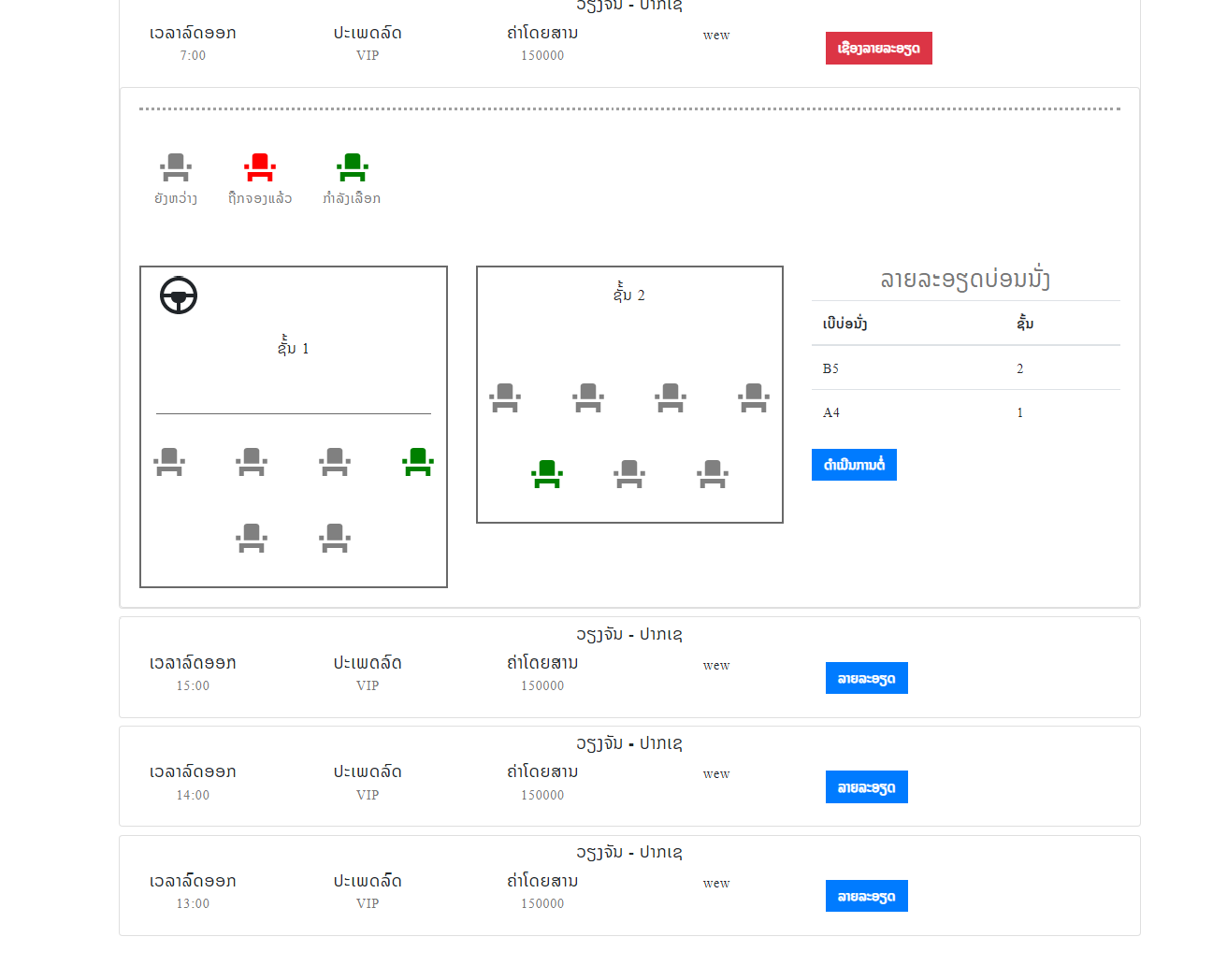
2

1

ຮູບທີ 9 : ຮູບແບບຟອມຄົ້ນຫາຖ້ຽວລົດ

1. ພາກສ່ວນຂໍ້ມູນຂອງຖ້ຽວລົດ.
2. ພາກສ່ວນປ້ອນຂໍ້ມູນຖ້ຽວລົດ.
3. ຄົ້ນຫາຖ້ຽວລົດ.

### 4.2.2 ຟອມເລືອກບ່ອນນັ່ງ



2

1

3

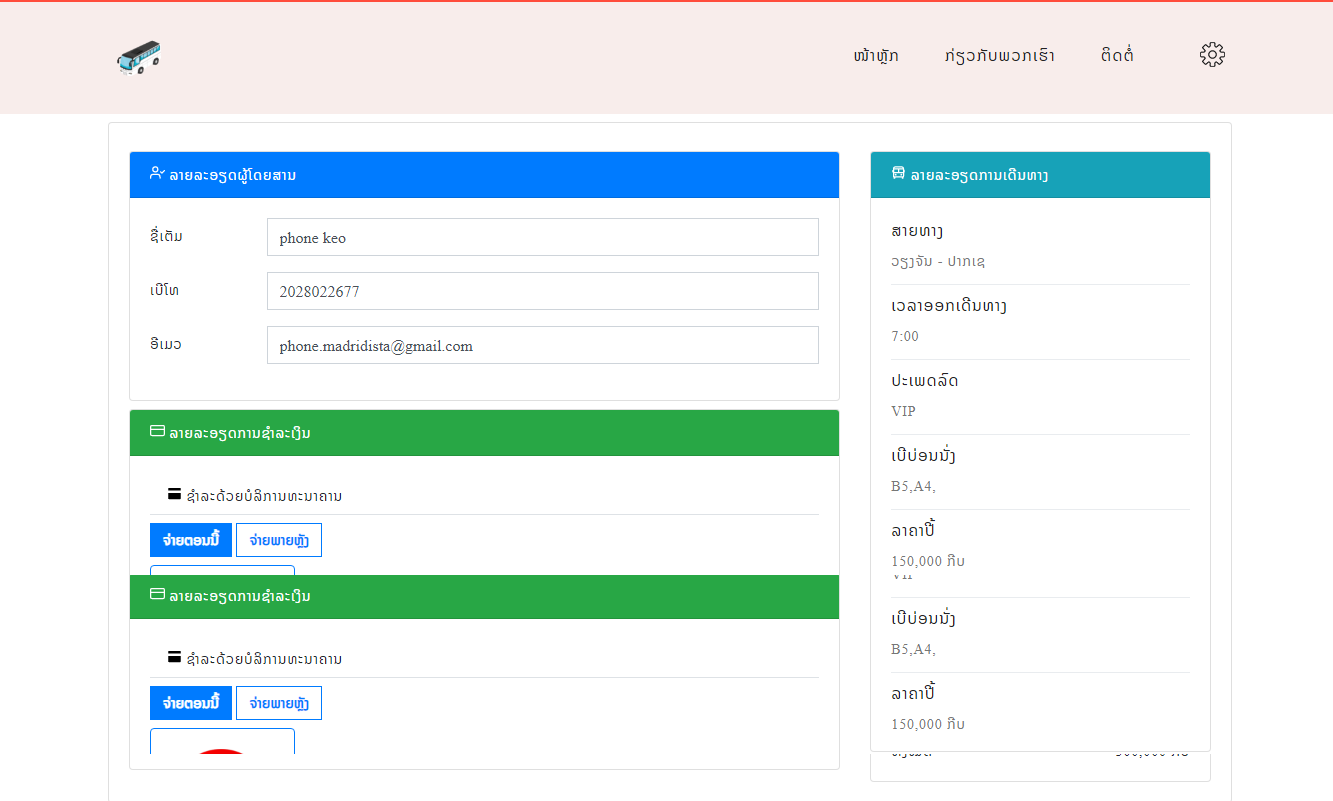
ຮູບທີ 10 : ຮູບແບບຟອມຈັດການຂໍ້ມູນບ່ອນນັ່ງ

1. ພາກສ່ວນຄຳອະທິບາຍຂໍ້ມູນບ່ອນນັ່ງ.
2. ລາຍລະອຽດບ່ອນນັ່ງທີ່ກຳລັງເລືອກ.
3. ດຳເນີນການຕໍ່.

### 4.2.3 ຟອມປ້ອນລາຍລະອຽດຜູ້ໂດຍສານ

2

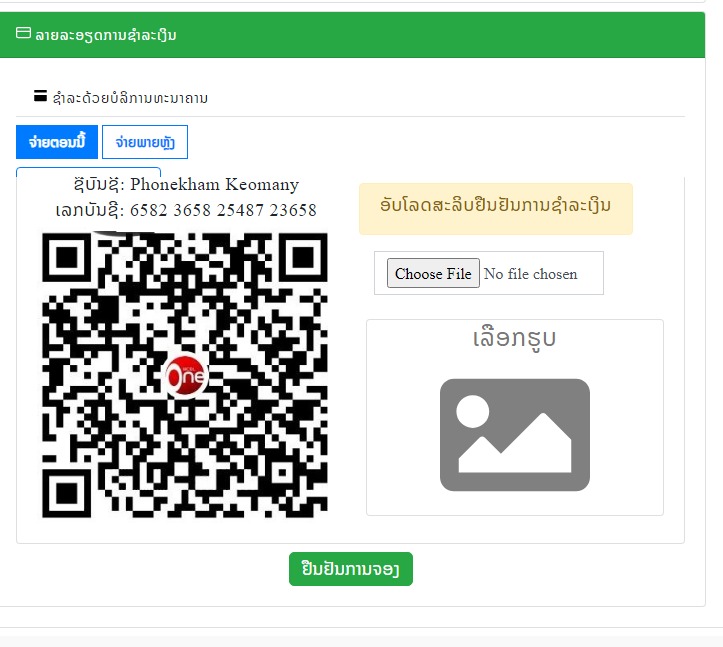
1

****

ຮູບທີ 11 : ຮູບແບບຟອມປ້ອນລາຍລະອຽດຜູ້ໂດຍສານ

1. ພາກສ່ວນປ້ອນຂໍ້ມູນຜູ້ໂດຍສານ.
2. ລາຍລະອຽດການຊໍາລະເງິນ.

### 4.2.4 ຟອມຊໍາລະເງິນ



3

2

1

ຮູບທີ 12 : ຮູບແບບຟອມຊໍາລະເງິນ

1. ເລືອກວ່າຈະຊຳລຕອນນີ້ ຫຼື ຈະຊຳລະພາຍຫຼັງ
2. ພາກສ່ວນອັບໂຫຼດສະລິບຢືນຢັນການຍຊໍາລະ.
3. ຢືນຢັນການຈອງ.

# **ບົດທີ 5**

# **ສະຫຼຸບ ແລະ ຂໍ້ສະເໜີ**

## 5.1 ສະຫຼຸບຜົນການຄົ້ນຄ້ວາ

ລະບົບຂາຍປີ້ລົດເມອອນລາຍຂອງສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້ ແມ່ນຖືກພັດທະນາຂຶ້ນເພື່ອຊ່ວຍໃຫ້ສະຖານນີມີລະບົບທີ່ທັນສະໄໝ, ສະດວກສະບາຍ ແລະ ເຮັດໃຫ້ການຈັດການຂໍ້ມູນ, ການບໍລິການພາຍໃນສະຖານນີຢ່າງວ່ອງໄວ, ມີຄວາມເປັນລະບຽບ, ສະດວກໃນການລາຍງານ, ຂໍ້ມູນທີ່ມີຄວາມຖືກຕ້ອງ ແລະ ຊັດເຈນ. ຂອບເຂດຂອງການດໍາເນີນວຽກງານໂປຣແກຣມຂອງພວກຂ້າພະເຈົ້າມີຈັດການຂໍ້ມູນ, ສະໝັກສະມາຊິກ, ບໍລິການ ແລະ ລາຍງານ. ເຊິ່ງເປັນການທົດແທນການເຮັດວຽກໃນລະບົບເກົ່າໂດຍລະບົບໃໝ່ ເພື່ອຫຼຸດຜ່ອນບັນຫາການເສຍຫາຍຂອງຂໍ້ມູນ. ພວກຂ້າພະເຈົ້າສາມາດພັດທະນາໂປຣແກຣມເພື່ອຊ່ວຍໃຫ້ເຮັດວຽກສະດວກສະບາຍຍິ່ງຂື້ນ ແລະ ໄດ້ອອກແບບໜ້າຟອມການປ້ອນຂໍ້ມູນຕ່າງໆ.

- ສ້າງຟອມຈັດການໄດ້ 5 ຟອມ.

- ຟອມການຄົ້ນຫາໄດ້ 5 ຟອມ.

- ພິມລາຍງານທັງໝົດໄດ້ 6 ລາຍງານ.

## 5.2 ຈຸດດີ

- ຮູ້ວິເຄາະຫາບັນຫາ ແລະ ສາເຫດຂອງລະບົບເກົ່າ.

- ຈະໄດ້ລະບົບຈອງປີ້ລົດເມສາຍໃຕ້ແບບອອນລາຍໃໝ່.

- ສາມາດນໍາໃຊ້ເວບໄຊທີ່ສ້າງຂຶ້ນມາເຂົ້າຊ່ວຍໃນການຈອງປີ້ລົດເມແບບອອນລາຍ.

- ເວບໄຊທີ່ສ້າງຂຶ້ນມາສາມາດຈອງອອນລາຍໄດ້.

- ການເຮັດບົດລາຍງານສະດວກສະບາຍ ແລະ ວ່ອງໄວຂື້ນ.

## 5.3 ຈຸດອ່ອນ

ໂປຣແກຣມນີ້ຍັງບໍ່ສົມບູນແລະຄົບຖ້ວນຕາມຄວາມຕ້ອງການຂອງສະຖານນີຂົນສົ່ງໂດຍສານສາຍໃຕ້ເທື່ອ ເນື່ອງຈາກວ່າໃນການສຶກສາຍັງບໍ່ທັນມີປະສົບການໃນການຂຽນເວບໄຊມາກ່ອນ, ຍັງບໍ່ຄວບຄຸມເຖິງຄວາມຕ້ອງການຕົວຈິງຂອງຜູ້ໃຊ້ລະບົບ.

## 5.4 ແນວທາງໃນການພັດທະນາ ແລະ ຂະຫຍາຍຕໍ່ຂອງສະຖານນີ

ເນື່ອງຈາກວ່າໂປຣແກຣມນີ້ເປັນໂປຣແກຣມທີ່ສ້າງຂື້ນມາແລ້ວ ດັ່ງນັ້ນ ເພື່ອເປັນການກວດສວບຫາຂໍ້ພິດພາດ ແລະ ເພື່ອຫາຈຸດດີຈຸດອ່ອນມາທຳການປັບປຸງ, ແກ້ໄຂຈຶ່ງຄວນນຳເອົາໂປຣແກຣມນີ້ໄປປັບປຸງຈຸດບົກຜ່ອງຂອງໂປຣແກຣມໃນບາງສ່ວນເພື່ອເຮັດໃຫ້ໂປຣແກຣມມີປະສິດທິພາບ ແລະ ເຮັດວຽກໄດ້ດີຂື້ນເພື່ອຈະນຳເອົາໄປປະກອບສ່ວນເຂົ້າໃນການເຮັດວຽກງານຕົວຈິງ.

ດັ່ງນັ້ນ ໂປຣແກຣມນີ້ເປັນໂປຣແກຣມໜຶ່ງທີ່ມີຄຸນສົມບັດໃນການໃຊ້ຖານຂໍ້ມູນໄດ້.

**ເອກະສານອ້າງອີງ**

[1] ທ້າວ ສາຍຄຳ ຊົ່ງວືຊົງ, ທ້າວຊືມົວ ຢົງຈື້ (2019-2020) ລບົບຂາຍປີ້ອອນໄລຂອງສະຖານີຂົນສົ່ງໂດຍສານສາຍໃຕ້

**ປະຫວັດຫຍໍ້ຜູ້ຂຽນບົດ**



ຊື່ ແລະ ນາມສະກຸນ: ທ້າວ ມະໂນພອນ ມະໂນກຸນ

ວັນ,ເດືອນ, ປີເກີດ: 16 ເດືອນ ທັນວາ ປີ 1995

ບ້ານເກີດ: ທົ່ງຂັນຄຳ ເມືອງ ຈັນທະບູລີ ແຂວງ ນະຄອນຫຼວງວຽງຈັນ

ບ້ານຢູ່ປັດຈຸບັນ: ທົ່ງຂັນຄຳ ເມືອງ ຈັນທະບູລີ ແຂວງ ນະຄອນຫຼວງວຽງຈັນ

ການສຶກສາ: ປີ 2016 ຈົບຊັ້ນສູງ ທີ່ ສະຖາບັນພັດທະນາສີມືແຮງງານລາວ - ເກົາຫຼີ

ປີ 2013 ຈົບມັດທະຍົມຕອນປາຍ: ທີ່ ມ.ສ ເຈົ້າອານຸວົງ

ປີ 2010 ຈົບມັດທະຍົມຕອນຕົ້ນ: ທີ່ ມ.ສ ເຈົ້າອານຸວົງ

ປີ 2007 ຈົບປະຖົມສົມບູນ: ທີ່ ໂຮງຮຽນປະຖົມສົມບູນອານຸ

ເບີໂທ: 020 5400 0003 , 020 2814 9278

ອີເມວ: [dou](mailto:lakitptt@gmail.com)\_2020@hotmail.com

ລາຍເຊັນເຈົ້າຂອງປະຫວັດ

ຊື່ແຈ້ງ..........................................

**ປະຫວັດຫຍໍ້ຜູ້ຂຽນບົດ**

ຊື່ ແລະ ນາມສະກຸນ: ທ້າວ ພອນຄຳ ແກ້ວມະນີ

ວັນ,ເດືອນ,ປີເກີດ: 07 ເດືອນ ມີນາ ປີ 1995

ບ້ານເກິດ:ບ້ານ ທ້ຽວ, ເມືອງ: ວຽງພູຄາ, ແຂວງ: ຫຼວງນໍ້າທາ ບ້ານຢູ່ປັດຈຸບັນ:ບ້ານ ທ້ຽວ, ເມືອງ: ວຽງພູຄາ, ແຂວງ: ຫຼວງນໍ້າທາ

ການສຶກສາ: ປີ 2016 ຈົບຊັ້ນສູງ: ທີ່ ສະຖາບັນພັດທະນາສີມືແຮງງານລາວ - ເກົາຫຼີ

ປີ 2013 ຈົບມັດທະຍົມສົມບູນ: ທີ່ ເມືອງວຽງພູຄາ

ປີ 2007 ຈົບປະຖົມສົມບູນ: ທີ່ ໂຮງຮຽນປະຖົມສົມບູນບ້ານດົງວຽງ

ເບີໂທ: 020 2802 2677

ອີເມວ: [phonekham.dev](mailto:lakitptt@gmail.com)@gmail.com

ລາຍເຊັນເຈົ້າຂອງປະຫວັດ

ຊື່ແຈ້ງ..........................................