**Location-based Facebook Chatbot**

Table of Contents

[**A.** **Introduction** 5](#_Toc492891098)

[1. Project information 5](#_Toc492891099)

[4. Problem definition 5](#_Toc492891100)

[5. Proposed solution 5](#_Toc492891101)

[5.1. Feature functions 5](#_Toc492891102)

[5.2. Advantages 5](#_Toc492891103)

[5.3. Disadvantages 5](#_Toc492891104)

[6. Functional Requirements 5](#_Toc492891105)

[7. Roles and Responsibilities 5](#_Toc492891106)

[**B.** **Software Project Management Plan** 5](#_Toc492891107)

[1. Problem definition 5](#_Toc492891108)

[1.1. Name of this project 5](#_Toc492891109)

[1.2. Problem abstract 5](#_Toc492891110)

[1.3. Project overview 5](#_Toc492891111)

[1.3.1. Current situation 5](#_Toc492891112)

[1.3.2. The proposed system 5](#_Toc492891113)

[1.3.3. Boundaries of the system 5](#_Toc492891114)

[1.3.4. Future plans 5](#_Toc492891115)

[1.3.5. Development environment 5](#_Toc492891116)

[1.3.5.1. Hardware requirement 5](#_Toc492891117)

[1.3.5.2. Software requirement 5](#_Toc492891118)

[2. Project organization 6](#_Toc492891119)

[2.1. Software process model 6](#_Toc492891120)

[2.2. Roles and responsibilities 6](#_Toc492891121)

[2.3. Tools and techniques 6](#_Toc492891122)

[2.4. Dependencies 6](#_Toc492891123)

[3. Project Management plan 6](#_Toc492891124)

[3.1. Tasks 6](#_Toc492891125)

[3.1.1. Initializing 6](#_Toc492891126)

[3.1.2. Planning 6](#_Toc492891127)

[3.1.3. Specifying requirements 6](#_Toc492891128)

[3.1.4. Designing database 6](#_Toc492891129)

[3.1.5. Create software design documents 6](#_Toc492891130)

[3.1.6. Implementing 6](#_Toc492891131)

[3.1.7. Testing 6](#_Toc492891132)

[3.1.8. Creating user’s manual 6](#_Toc492891133)

[3.2. Tasks sheet 6](#_Toc492891134)

[3.3. All meeting minutes 6](#_Toc492891135)

[4. Coding convention 6](#_Toc492891136)

[**C.** **Software Requirement Specification** 6](#_Toc492891137)

[1. User requirement specification 6](#_Toc492891138)

[1.1. Guest requirement 6](#_Toc492891139)

[1.2. User requirement 6](#_Toc492891140)

[1.3. Administrator requirement 6](#_Toc492891141)

[2. System requirement specification 6](#_Toc492891142)

[2.1. External interface requirement 6](#_Toc492891143)

[2.1.1. User interface 6](#_Toc492891144)

[2.1.2. Hardware interface 6](#_Toc492891145)

[2.1.3. Software interface 6](#_Toc492891146)

[2.1.4. Communication protocol 6](#_Toc492891147)

[2.2. System overview use case 6](#_Toc492891148)

[2.3. Use cases 6](#_Toc492891149)

[3. Software system attribute 6](#_Toc492891150)

[3.1. Usability 6](#_Toc492891151)

[3.1.1. Graphic user interface 6](#_Toc492891152)

[3.1.2. Usability 6](#_Toc492891153)

[3.1.3. Installation 6](#_Toc492891154)

[3.2. Reliability 6](#_Toc492891155)

[3.3. Availability 6](#_Toc492891156)

[3.4. Security 7](#_Toc492891157)

[3.5. Maintainability 7](#_Toc492891158)

[3.6. Portability 7](#_Toc492891159)

[3.7. Performance 7](#_Toc492891160)

[4. Conceptual program 7](#_Toc492891161)

[**D.** **Software Design Description** 7](#_Toc492891162)

[1. Design overview 7](#_Toc492891163)

[2. System architecture design 7](#_Toc492891164)

[2.1. Web application architecture description 7](#_Toc492891165)

[3. Component diagram 7](#_Toc492891166)

[4. Detailed description 7](#_Toc492891167)

[4.1. Class diagram 7](#_Toc492891168)

[4.2. Class diagram explanation 7](#_Toc492891169)

[4.3. Interactive diagram 7](#_Toc492891170)

[5. User interface design 7](#_Toc492891171)

[6. Database design 7](#_Toc492891172)

[6.1. Entity relationship diagram 7](#_Toc492891173)

[6.2. Entity dictionary 7](#_Toc492891174)

[6.2.1. AspNetUsers 8](#_Toc492891175)

[6.2.2. AspNetUserClaims 8](#_Toc492891176)

[6.2.3. AspNetUserLogins 8](#_Toc492891177)

[6.2.4. AspNetUserRoles 8](#_Toc492891178)

[6.2.5. AspNetRoles 8](#_Toc492891179)

[6.2.6. Campaign 8](#_Toc492891180)

[6.2.7. Combo 8](#_Toc492891181)

[6.2.8. ComboProductMap 8](#_Toc492891182)

[6.2.9. Corporation 8](#_Toc492891183)

[6.2.10. Coupon 8](#_Toc492891184)

[6.2.11. ImageLink 8](#_Toc492891185)

[6.2.12. Order 8](#_Toc492891186)

[6.2.13. OrderDetail 8](#_Toc492891187)

[6.2.14. Product 8](#_Toc492891188)

[6.2.15. ProductCategory 8](#_Toc492891189)

[6.2.16. ProductCatgoryMap 8](#_Toc492891190)

[6.2.17. Promotion 8](#_Toc492891191)

[6.2.18. Shop 8](#_Toc492891192)

[6.2.19. ShopCategory 8](#_Toc492891193)

[6.2.20. ShopCategoryMap 8](#_Toc492891194)

[7. Algorithms 8](#_Toc492891195)

1. **Introduction**
   1. Project information
      * Project name: **Location-based Facebook Chat bot**
      * Project code: **LBFC**
      * Product type: **Facebook messenger bot + Web application**
      * Start date: **September 4th, 2017**
      * End date: **December 12th, 2017**
   2. Introduction

In this document, we will introduce the Location-based Facebook Chat bot solution, which helps customers shopping easier without any required app but Facebook messenger – an app that every Facebook user need to install. Besides, it can help shops admin manage products and other services such as promotions or coupons and orders.

* 1. Current situation

When a shop or a company want to promote their products and services, a mobile app is needed since a web page on smartphone can’t actually replace all the functions in a desktop web page better than a mobile app. However, many companies are now creating their own apps. This can make users feel uncomfortable since they have to install an app whenever they visit a new store, which can make their phones slow down.

* 1. Problem definition

Current problems:

* **Too many steps to install an app**: To install an app, users have to process a lot of steps to get app installed. On the other hand, more apps installed means less storage available on the phone. Therefore, people prefer less app but with same functionality.
* **Website for mobile can’t display all functions**: Because of the small screen, a website for mobile needs to reduce all details to leave more space for primary functions. This lead to lack of functionality. If the webpage decides not to reduce then the user experience will become badly.
  1. Proposed solution

Our solution is Location-based Facebook Chat bot. This bot will help customers explore shops just by chatting. From product information to promotion information and ordering, this bot can automatically do that without any click. Furthermore, it can detect customer location via Wi-Fi and recommend the services in the shop and surrounding stores. Customer can also get coupon or promotion from this bot.

* + 1. Feature functions
* **Auto-detect customer location:** When a customer visits a shop, connect to Wi-Fi or enter a code from his/her invoices, system automatically detects where the customer is and process to another functions based on the location.
* **Recommend products and promotions:** Based on the location, system will search for products of the current shop and surrounding shops, also the promotions.
* **Allow customers to order:** customers can order products in the current shop or nearby shops. System will return an order code for each customer to track their orders.
* **Real-time notifications:** notifications about services of the current shop and nearby shops will be send in real-time. When a shop update its services, users will be notified immediately.
* **Service management:** Staffs from all the shop associated with system can control the orders and services via one site only.
  + 1. Advantages
* **Simple to use:** customers don’t need to install any app to use this system. All customers need is to install Facebook messenger, which is really popular since it is another must-installed for Facebook user.
* **Customers don’t have to research for places:** information of shops are all included in the system, therefore users don’t have to search around to get information about services of shops nearby. Besides, the system is in real-time so customers don’t have to refresh anything and the bot will automatically advice new services.
* **Customers can get coupon on their own phone:** Usually customers have to come to a store to receive coupon. In this system, a coupon is presented as code and users only have to copy the code for the bot. Then, coupon can be used.
* + 1. Disadvantages
* **A bot can’t replace a human:** An AI bot can’t replace a human totally and it still need some supporters in case of strange behaviors from customers.
* **Some business won’t suit for all brands:** because the management site is created for all brands, it can’t meet some special requirements of some brands.
  1. Functional Requirements
     1. For guests
* Order:
  + See full list of menu of a shop
  + Create order
  + See full list of order detail
  + Cancel order
* Promotion and Coupons:
  + Get coupons and promotions
  + Apply coupons and promotions in order
  + See list of available coupons and promotions
* Search for nearby shops
  + 1. For shop staffs
* Manage products
  + Add product to inventory
  + Update product information
  + Get product information
  + Delete product
* Manage promotions
  + Add promotions
  + Get promotions information
  + Update promotions information
  + Delete promotion
  + Deactivate promotions
* Manage orders
  + Get all orders information in shop
  1. Roles and Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Name | Role | Position | Contact |
| 1. | Tường Đức Khang | Developer | Leader | [khangtdse61752@fpt.edu.vn](mailto:khangtdse61752@fpt.edu.vn) |
| 2. | Thái Hoàng Duy Bảo | Developer | Member | baothdse |
| 3 | Nguyễn Văn Hùng | Developer | Member | hungnvse61305@fpt.edu.vn |
| 4. | Dương Hải Quang | Developer | Member | quangdhse61383@fpt.edu.vn |

1. **Software Project Management Plan**
   1. Problem definition
      1. Name of this project
      2. Problem abstract
      3. Project overview
         1. Current situation
         2. The proposed system
         3. Boundaries of the system
         4. Future plans
         5. Development environment
            1. Hardware requirement
            2. Software requirement
   2. Project organization
      1. Software process model
      2. Roles and responsibilities
      3. Tools and techniques
      4. Dependencies
   3. Project Management plan
      1. Tasks
         1. Initializing
         2. Planning
         3. Specifying requirements
         4. Designing database
         5. Create software design documents
         6. Implementing
         7. Testing
         8. Creating user’s manual
      2. Tasks sheet
      3. All meeting minutes
   4. Coding convention
2. **Software Requirement Specification**
   1. User requirement specification
      1. Guest requirement
      2. User requirement
      3. Administrator requirement
   2. System requirement specification
      1. External interface requirement
         1. User interface
         2. Hardware interface
         3. Software interface
         4. Communication protocol
      2. System overview use case
      3. Use cases
   3. Software system attribute
      1. Usability
         1. Graphic user interface
         2. Usability
         3. Installation
      2. Reliability
      3. Availability
      4. Security
      5. Maintainability
      6. Portability
      7. Performance
   4. Conceptual program
3. **Software Design Description**
   1. Design overview
   2. System architecture design
      1. Web application architecture description
   3. Component diagram
   4. Detailed description
      1. Class diagram
      2. Class diagram explanation
      3. Interactive diagram
   5. User interface design
   6. Database design
      1. Entity relationship diagram
      2. Entity dictionary

|  |  |
| --- | --- |
| Entity name | Description |
| AspNetUsers | Storing user information. User can be admin or shop staffs, not guest. |
| AspNetUserClaims |  |
| AspNetUserLogins | Storing information of 3rd party login provider, such as Google, Facebook. |
| AspNetUserRoles | Mapping between users and roles |
| AspNetRoles | Storing roles |
| Campaign | Represent a promotion campaign |
| CampaignShopMap | Mapping between campaign and shop, since one shop can has more than one campaign and vice versa. |
| Combo | Represent a combo of products |
| ComboDetail | Detail of combo, mapping between combo and product, tells which product is in which combo |
| Corporation | Represent a brand |
| Coupon | Storing coupons information |
| ImageLink | Storing image link for product, shop. |
| Order | Storing order |
| OrderDetail | Storing details of orders |
| Product | Represent a product |
| ProductShopMap | Storing which product in which shop |
| ProductCategory | Represent categories of products |
| ProductCatgoryMap | Mapping between products and products categories |
| Promotion | Represent a promotion |
| Shop | Represent a shop |
| ShopCategory | Represent a shop category |
| ShopCategoryMap | Mapping between shop and category |

* + - 1. AspNetUsers

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | nvarchar(128) | Primary key, Not null | Unique key for this table |
| Email | nvarchar(256) | null | users emails |
| EmailConfirmed | bit | not null | check whether user email is confirmed or not |
| PasswordHash | nvarchar(MAX) | null | user password |
| SecurityStamp | nvarchar(MAX) | null | security token for login without re-enter password |
| PhoneNumber | nvarchar(MAX) | null | user phone number |
| PhoneNumberConfirmed | bit | not null | check whether user phone number is confirmed or not |
| TwoFactorEnabled | bit | not null | check whether user is using two factor security |
| LockoutEndDateUTC | datetime | null |  |
| LockoutEnable | bit | not null |  |
| AccessFailedCount | int | not null | number of attempt login failed |
| Username | nvarchar(256) | not null | username of user |

* + - 1. AspNetUserClaims
      2. AspNetUserLogins

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| LoginProvider | nvarchar(128) | primary key, not null | Name of 3rd party login provider |
| ProviderKey | nvarchar(128) | primary key, not null | Key that provider provided |
| UserId | nvarchar(128) | primary key, not null,  foreign key to AspNetUsers table | User Id |

* + - 1. AspNetUserRoles

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| UserId | nvarchar(128) | primary key, not null,  foreign key to AspNetUsers | User Id |
| RoleId | nvarchar(128) | primary key, not null,  foreign key to AspNetRoles | Id of the specified role |

* + - 1. AspNetRoles

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | nvarchar(128) | primary key, not null | Unique Id for this table |
| Name | nvarchar(128) | not null | Name of the role |

* + - 1. Campaign

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | int | primary key, not null, auto increment | Unique Id for this table |
| Name | nvarchar(128) | not null | Name of campaign |
| StartDate | datetime | null | The date when the campaign started |
| EndDate | datetime | null | The date when the campaign ended |
| IsActive | bit | not null | The campaign is active or not |

* + - 1. CampaignShopMap

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| CampaignId | int | primary key, not null, foreign key to Campaign | Id of campaign |
| ShopId | int | primary key, not null, foreign key to Shop | Id of shop |

* + - 1. Combo

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | int | primary key, not null, auto increment | Unique Id of this table |
| Name | nvarchar(128) | not null | Name of this combo |
| ShopId | int | not null, foreign key to Shop | Id of shop which has this combo |
| Price | money | null | Price for this combo (usually combo has lower price than individuals) |
| IsActive | bit | not null | Check if this combo is active or not |

* + - 1. ComboDetail

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| ComboId | int | primary key, not null, foreign key to Combo | Id of combo |
| ProductId | int | primary key, not null, foreign key to Product | Id of product |
| Quantity | int | not null | Quantity of a product in a combo |
| IsActive | bit | not null | Check if this is active or not |

* + - 1. Corporation

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | int | primary key, not null, auto increment | Unique Id of this table |
| Name | nvarchar(128) | not null | Name of a corporation |
| Description | nvarchar(MAX) | null | Description for corporation |
| IsActive | bit | not null | Check if this corporation is active or not |

* + - 1. ImageLink

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | int | primary key, not null, auto increment | Unique Id of this table |
| ShopId | int | null | Id of shop |
| ProductId | int | null | Id of product |
| Link | varchar(256) | not null | Link of image |
| IsActive | bit | not null | Check if this link is active or not |

* + - 1. Order

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | int | primary key, not null, auto increment | Unique Id of this table |
| CustomerName | nvarchar(50) | not null | Name of customer |
| CustomerAddress | nvarchar(256) | not null | Address of customer |
| CustomerPhone | varchar(16) | not null | Phone number of customer |
| DateOrdered | datetime | not null | Date of order |
| PromotionId | int | null | Id of promotion (if has) |
| StaffReceived | nvarchar(128) | null | Id of staff who receives the order |
| Status | smallint | not null | Status of the order (Waiting, Delivering, Done) |
| IsActive | bit | not null | Check if this order is active or not |

* + - 1. OrderDetail

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | int | primary key, not null, auto increment | Unique Id of this table |
| OrderId | int | not null | Id of order |
| ProductId | int | not null | Id of product |
| Quantity | int | not null | Quantity of this product |
| CurrentPrice | money | not null | Current price of a unit of this product (sometimes current price can be lower than usual price due to promotions) |
| PromotionId | int | null | Id of promotion (if has) |
| IsActive | bit | not null | Check if this order detail is active or not |

* + - 1. Product

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Type | Other constraints | Description |
| Id | int | primary key, not null, auto increment | Unique Id of this table |
| Name | int | not null | Name of this product |
| Price | money | not null | Price of this product |
| Description | nvarchar(MAX) | null | Description of this product |
| ShopId | int | not null | Id |
| PromotionId | int | null | Id of promotion (if has) |
| IsActive | bit | not null | Check if this order detail is active or not |

* + - 1. ProductCategory
      2. ProductCatgoryMap
      3. Promotion
      4. Shop
      5. ShopCategory
      6. ShopCategoryMap
  1. Algorithms
     1. Vietnamese converter
        1. Introduction

Since this is a chat bot application, it has to deal with human and not all the time that a person can chat with the correct grammars, words. There are also a lot of slangs in Vietnamese that doesn’t appear in formal textbook. Because of that, this chat bot application has to have the ability to translate some slangs or acronyms in speaking language to correct grammar and words in order to understand.

* + - 1. Common approaches

There are two common approaches: grammar-based and statistical language models. In grammar-based, a sentence is separated into words labeled with types. For example, “tôi muốn uống trà sữa” will be separated to:

“tôi” - Noun  
“muốn” – Verb  
“uống” –Verb  
“trà sữa” – Noun

System will analyze this sentence and store as a sentence with “N-V-V-N” grammar model. The system will analyze others sentences whether they match stored grammar models and correct. Another approach, statistical language models, is used more widely. It will analyze a sentence and see which word usually appears after another word. For example, in the old sentence “tôi muốn uống trà sữa”, it will return a result that: “muốn” is usually after “tôi”, “uống” is usually after “muốn” and so on and it will correct based on this chain of words.

* + - 1. Solution design

Because we will only deal with people who want to buy some products – who usually say simply, not a writer or language professor – whose sentences are usually more difficult for a machine to understand, we will choose the statistical language model approach for easier implementation but the accurate will stay high enough. In statistical language model, we will use N-gram algorithms to calculate the statistic.

* + - * 1. N-gram

N-gram is the common algorithms used for counting words and calculating relationship of them based on listed sentences. As a result, a sentence input by a user will be calculate for probability and defined whether it is correct or not. The probability of a sentence consists of n words is calculated as:

P(s) = P(w1) \* P(w2) \* …\* P(wn)

This will lead to another task is to calculate probability of the upcoming word, since in the above formula, P(w2) is not independent from any previous word w1.

We will use the chain rule to expand the previous formula.

This is impossible since it needs a very enormous data that we will never have enough to count for possible sentences since n could be very large.

* + - * 1. Markov assumption

In other words, we approximated each component in the product

Our remaining issue is to determine the number k. With k = 0, we have unigram; k = 1, we have bi-gram, k = 2, we have tri-gram and so on. The larger the number k, the more accurate it can translate. However, since we only need to deal with normal conversation, not complicated textbook or letter so in practice, we chose the bi-gram model since unigram tri-gram or above will be highly complicated and is not necessary for our solution.

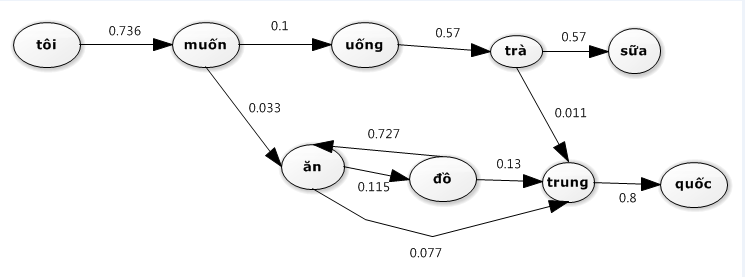
For example, in bi-gram, we have a sentence “tôi muốn uống trà sữa”

With real statistic from our training data, the probability of that sentence is

P (tôi muốn uống trà sữa) = 0.789 \* 0.736 \* 0.1 \* 0.571 \* 0.571 = 0.019

* + - * 1. Graph

Because we calculated based on groups of two-word, we can present each word as a vertex in a graph of words. We can’t use a list since it can’t present the characteristic of each word – it has a lot of links to another words to create a sentence. The probabilities we calculated before is now edges of the graph. Based on the probabilities, if there are more than one corrected sentence, it will choose the sentence with more probability. They also useful when searching for words, since the words with higher probabilities will be searched first.



* + 1. Bot