LIVE BUDDY

GO FINAL PROJECT

Group 7 :

黃淯琪 111511157 / 林芷卉 111511102 / 廖鉑涵 111511146 / 賴柏諺 111511071

Content

- Motivation
- Function
- Code
- Demo

MOTIVATION

Function

- Query the weather
- Scheduled Weather Updates
- Query the exchange rate
- Query the trending news
- Translation function
- Task reminder

IMPLEMENTATION

Variable

```
var(
    bot *linebot.Client
    tasks = make(map[string][]Task)
    userIDs = make(map[string]bool)  // 可紀錄多使用者，使用 map 儲存避免重複
    api_weather = "307d7cb0b615c6226d4c9d384bf78d92"
    api_rate = "94b94e5943b3454087fd8fc3"
    api_news = "3da1dcdd55644f1ea5e650cd7c70d7e0"
)
```

Initialize server

```go
func main() {
    // 初始化 Line Bot
    // channel secret // channel token
    var err error
    CHANNEL_SECRET := "f66aa5c243e3e7216400f93fad900592"
    CHANNEL_ACCESS_TOKEN := "StvJ0CKW2p91RWI64mxtNdlxOCQGsJA1JrU5AV5ZdGUqs7QGzgjEM2X
    bot, err = linebot.New(CHANNEL_SECRET, CHANNEL_ACCESS_TOKEN)
    if err != nil {
        log.Fatal(err)
    }

    // 設置 HTTP Handler
    http.HandleFunc("/callback", callbackHandler)

    // 啟動定時功能，城市預設為新竹
    go func() {
        for range time.Tick(1 * time.Hour) { // 若要測一分鐘發送一次改為time.Minute
            sendWeatherNotification("Hsinchu")
        }
    }()

    // 啟動伺服器
    log.Println("Line bot server started at :8080")
    if err := http.ListenAndServe(":8080", nil); err != nil {
        log.Fatal("ListenAndServe:", err)
    }
}
```

Store ID and response

- **負責處理Line平台的訊息回應**
- **針對不同的關鍵字呼叫不同的** function

```
163 func callbackHandler(w http.ResponseWriter, req *http.Request) {
164     events, err := bot.ParseRequest(req)
165     if err != nil {
166         w.WriteHeader(http.StatusBadRequest)
167         return
168     }
169     for _, event := range events {
170         // 獲取 userID
171         userID := event.Source.UserID
172         if _, exists := userIDs[userID]; !exists {···
175         }
176         if event.Type == linebot.EventTypeMessage {
177             switch message := event.Message.(type) {
178             case *linebot.TextMessage:
179                 if message.Text == "天氣" {···
183                 } else if message.Text == "匯率" {···
187                 } else if message.Text == "開啟" {
188
189                 } else if message.Text == "新聞" {···
193                 } else if message.Text == "翻譯" {···
197                 } else if message.Text == "任務" {···
201                 } else if message.Text == "完成" {···
205                 } else if strings.HasPrefix(message.Text, "城市：") {···
212                 } else if strings.Contains(message.Text, "to") {···
235                 } else if strings.HasPrefix(message.Text, "新聞：") {···
247                 } else if strings.HasPrefix(message.Text, "翻譯：") {···
252                 } else if strings.HasPrefix(message.Text, "任務：") {···
258                 } else if message.Text == "任務清單" {···
264                 } else if strings.HasPrefix(message.Text, "完成：") {···
271                 } else {
```

Query the weather

輸入格式 「城市：城市名稱」

例如：　城市：新竹　、　城市：台北

回傳：
天氣狀況、氣溫、濕度、降雨量、降雨機率

Query the weather

用天氣 API 取得
公開氣象資訊

```go
// 獲取即時天氣資料
Tabnine | Edit | Test | Explain | Document | Ask
func getWeather(apiKey, city string) string {
    // 設置天氣 API 請求 URL
    url := "http://api.openweathermap.org/data/2.5/forecast?q=" + city + "&appid=" + apiKey + "&units=metric"
    resp, err := http.Get(url)
    if err != nil || resp.StatusCode != http.StatusOK {
        return "無法獲取天氣資料，請檢查城市名稱是否正確。"
    }
    defer resp.Body.Close()
    var weather WeatherResponse
    if err := json.NewDecoder(resp.Body).Decode(&weather); err != nil {
        return "解析天氣資料失敗"
    }
    return fmt.Sprintf(
        "城市: %s\n" +
        "天氣狀況: %s\n" +
        "當前氣溫: %.1f°C\n" +
        "體感溫度: %.1f°C\n" +
        "濕度: %v\n" +
        "過去三小時降雨量: %.2f mm\n" +
        "降雨機率: %.2f",
        city, weather.List[0].Weather[0].Description, weather.List[0].Main.Temp,
        weather.List[0].Main.FeelsLike, weather.List[0].Main.Humidity,
        weather.List[0].Rain.ThreeHour, weather.List[0].Pop)
}
```

Scheduled Weather Updates

- **用戶傳送任何一則訊息就會自動紀錄** userID
- **定時一小時傳送一次天氣資訊（預設新竹）**

  （測試時設定一分鐘傳送一次）
- **可同時傳送給多個用戶**

Scheduled Weather Updates

使用 GO 的 time 標準庫

```go
// 啟動定時功能，城市預設為新竹
go func() {
    for range time.Tick(1 * time.Hour) {
        sendWeatherNotification("Hsinchu")
    }
}()
```

```go
// 獲取 userID
userID := event.Source.UserID
if _, exists := userIDs[userID]; !exists {
    userIDs[userID] = true
    log.Printf("新增 userID: %s", userID)
}
```

```go
func sendWeatherNotification(city string) {
    weather := getWeather(api_weather, city)
    for userID := range userIDs {
        if _, err := bot.PushMessage(userID, linebot.NewTextMessage(weather)).Do(); err != nil {
            log.Printf("發送訊息給 userID %s 失敗: %v", userID, err)
        } else {
            log.Printf("成功發送訊息給 userID: %s", userID)
        }
    }
}
```

Query the exchange rate

輸入格式 「貨幣代碼 to 貨幣代碼」

例如： USD to TWD, JPY to USD

Query the exchange rate

用匯率 API 取得兩種貨幣間的匯率

```go
// 查詢即時匯率的函數
Tabnine | Edit | Test | Explain | Document | Ask
func getExchangeRate(fromCurrency, toCurrency, apiKey string) (float64, error) {
    // 建立 API 請求的 URL
    url := fmt.Sprintf("https://v6.exchangerate-api.com/v6/%s/latest/%s", apiKey, fromCurrency)
    // 發送 GET 請求
    resp, err := http.Get(url)
    if err != nil {
        return 0, fmt.Errorf("無法連接到匯率 API: %v", err)
    }
    defer resp.Body.Close()
    // 檢查是否成功獲得回應
    if resp.StatusCode != http.StatusOK {
        return 0, fmt.Errorf("API 回應錯誤: %s", resp.Status)
    }
    // 定義回應數據結構
    type ExchangeRateResponse struct {
        ConversionRates map[string]float64 `json:"conversion_rates"`
    }
    // 解析 JSON 回應資料
    var data ExchangeRateResponse
    if err := json.NewDecoder(resp.Body).Decode(&data); err != nil {
        return 0, fmt.Errorf("解析匯率資料失敗: %v", err)
    }
    // 查找目標貨幣的匯率
    if rate, exists := data.ConversionRates[toCurrency]; exists {
        return rate, nil
    }
```

Query the trending news

輸入格式 「新聞：搜尋關鍵字」

例如： 新聞：戒嚴

會將最近搜尋次數前五的新聞回傳

Query the trending news

```go
//查詢新聞並回傳flex message
func getNews(apiKey, keyword string) *linebot.FlexMessage {
    url := fmt.Sprintf("https://newsapi.org/v2/everything?q=%s&sortBy=popularity&apiKey=%s", keyword, apiKey)
    resp, err := http.Get(url)
    if err != nil || resp.StatusCode != http.StatusOK {
        log.Println("Unable to fetch news:", err)
        return nil
    }
    defer resp.Body.Close()

    var news NewsResponse
    if err := json.NewDecoder(resp.Body).Decode(&news); err != nil {
        log.Println("Failed to parse news data:", err)
        return nil
    }

    if len(news.Articles) == 0 {
        return nil
    }
```

flex message implement

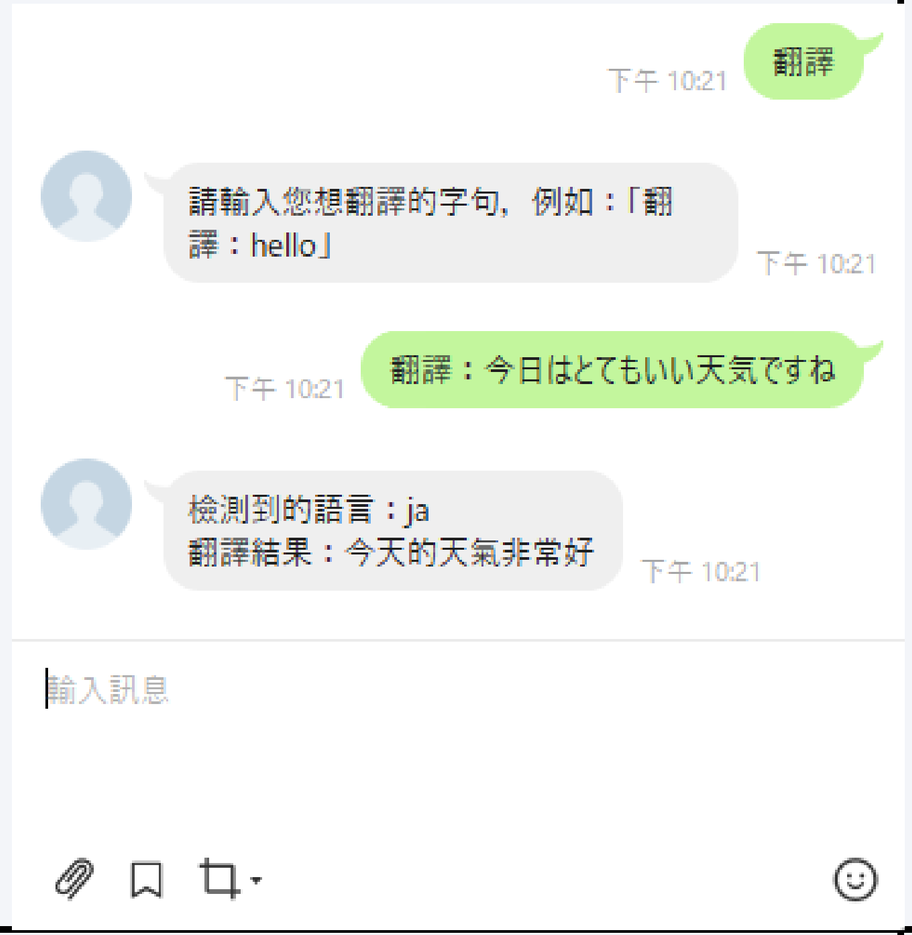
```
for i, article := range news.Articles {
    if i >= 5 {
        break // Only show the first 5 articles
    }

    // Title Component
    contents = append(contents, &linebot.TextComponent{
        Type:  "text",
        Text:  article.Title,
        Size:  "lg", // Large size for the tit
        Wrap:  true,
    })

    // Source Component
    contents = append(contents, &linebot.Text(
        Type:  "text",
        Text:  fmt.Sprintf("Source: %s", arti(
        Size:  "sm",  // Small size for the s(
        Wrap:  true,
    })

    // URL Link Component
    contents = append(contents, &linebot.Text(
        Type:  "text",
        Text:  fmt.Sprintf("Read More: %s", ar
        Size:  "sm",  // Small size for the l:
        Wrap:  true,
        Color: "#0066CC",  // Link color
        Action: &linebot.URIAction{
            Label: "Click Here",
            URI:   article.URL,
        },
    })
}
```

```
// Bubble Header
header := &linebot.TextComponent{
    Type:   "text",
    Text:   "Relative Trending News",
    Size:   "xl",  // Extra large size for the header
    Align:  "center",
}

// Bubble Body
body := &linebot.BoxComponent{
    Type:   "box",
    Layout: "vertical",
    Contents: contents,
}

// Wrap header in a BoxComponent
headerBox := &linebot.BoxComponent{
    Type:   "box",
    Layout: "horizontal",
    Contents: []linebot.FlexComponent{header},  // Wrap
}

// Flex Bubble
bubble := &linebot.BubbleContainer{
    Type:   "bubble",
    Header: headerBox,
    Body:   body,
}
```

Translation function

輸入格式 「翻譯：句子」

例如：翻譯：今日はとてもいい天気ですね
翻譯：Het is zo mooi weer vandaag



Translation function

```
575  func handleTranslation(input string) string {
576      // 檢查是否以 "翻譯：" 開頭
577      if !strings.HasPrefix(input, "翻譯：") {…
579      }
580      // 提取需要翻譯的文字
581      stringToTranslate := strings.TrimPrefix(input, "翻譯：")
582      // 設定 Translator API 詳細資料
583      transKey := "961Q45EeefSNS2E1xc4ouDAyKUJMCD0M4F7LJhpNaDkgbTFQQ3pmJQQJ99ALACxCCsyXJ3w3AAAbACOGtYa8"
584      transURL := "https://api.cognitive.microsofttranslator.com/translate"
585      params := "?api-version=3.0&to=zh-Hant"
586      fullURL := transURL + params
587      headers := map[string]string{…
591      }
592      // 準備請求體
593      body := []map[string]string{{"text": stringToTranslate}}
594      bodyJSON, err := json.Marshal(body)
595      if err != nil {
596          return fmt.Sprintf("JSON 序列化錯誤：%v", err)
597      }
598      // 建立 HTTP 請求
599      req, err := http.NewRequest("POST", fullURL, bytes.NewBuffer(bodyJSON))
600      if err != nil {
601          return fmt.Sprintf("請求建立錯誤：%v", err)
602      }
603      // 設置請求頭
604      for key, value := range headers {
605          req.Header.Set(key, value)
606      }
607      // 發送請求
608      client := &http.Client{}
609      resp, err := client.Do(req)
```

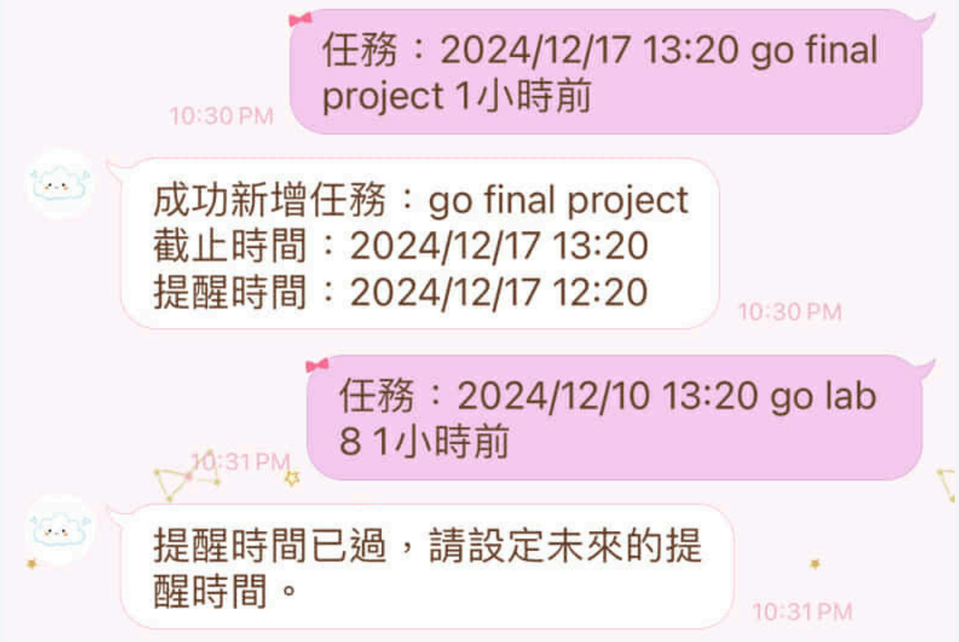
Translation function

```
610     if err != nil {
611         return fmt.Sprintf("HTTP 請求錯誤：%v", err)
612     }
613     defer resp.Body.Close()
614     // 讀取回應
615     responseBody, err := ioutil.ReadAll(resp.Body)
616     if err != nil {
617         return fmt.Sprintf("回應讀取錯誤：%v", err)
618     }
619     // 定義回應結構
620     var response []struct {
621         DetectedLanguage struct {
622             Language string `json:"language"`
623         } `json:"detectedLanguage"`
624         Translations []struct {
625             Text string `json:"text"`
626         } `json:"translations"`
627     }
628     // 解析 JSON 回應
629 >   if err := json.Unmarshal(responseBody, &response); err != nil {···
631     }
632     // 提取翻譯結果
633 >   if len(response) > 0 {···
640     }
641     return "未收到有效的翻譯回應。"
642 }
```

Task reminder - Add task

輸入格式 「任務：YYYY/MM/DD HH:MM TASK X**小時前」**

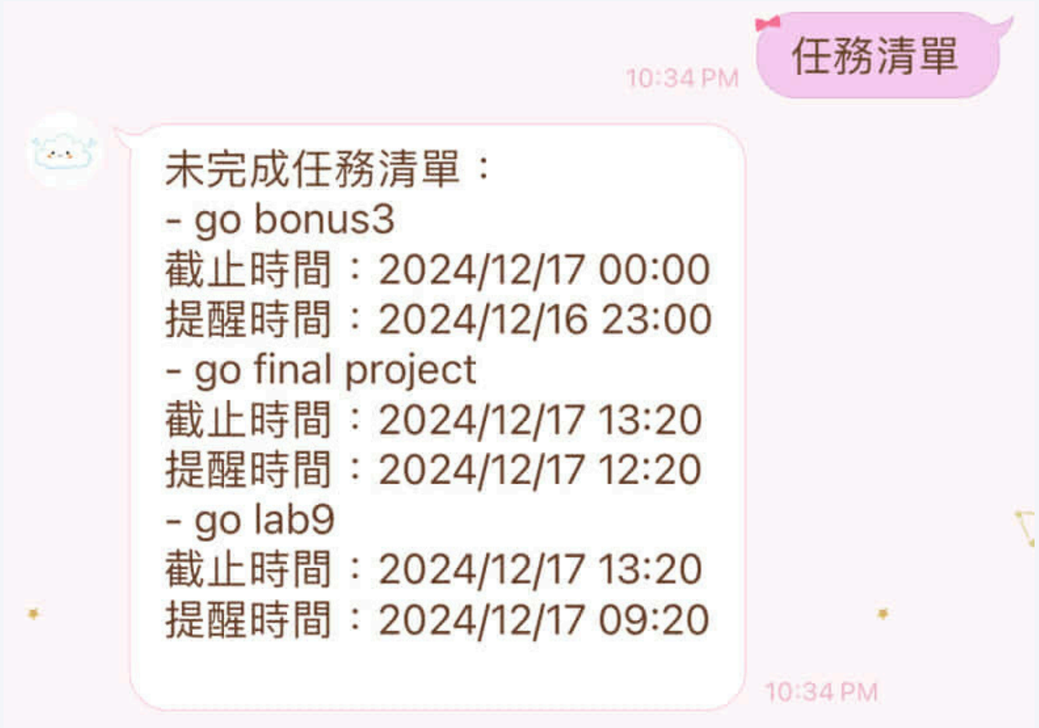
- 輸入任務指令來新增待辦事項與提醒設定
- 進行格式驗證
  - 如果提醒時間已過，系統會即時提示用戶重新設定
  - 如果格式輸入錯誤，系統會提醒正確格式並提示用戶重新設定
- 系統根據設定的提醒時間，自動通知用戶，確保不會錯過重要的ddl。
- 利用 Goroutine 後台處理提醒通知，提升系統效率。



Task reminder - List task

輸入格式 「任務清單」

- 任務過濾條件：
  - 未完成 (IsCompleted == false)
  - 截止時間 (Deadline) 未過期。
- 排序邏輯：
  - 根據任務的截止時間由早到晚排序。
- 輸出格式：
  - 若無任務或無符合條件的任務，回傳：「目前沒有任何未完成的任務。」
  - 否則，回傳格式化的任務清單：



Task reminder - Complete task

輸入格式 「完成：TASK」

- 功能邏輯：
  - 確認該使用者是否存在任務。
  - 將輸入的任務名稱進行標準化處理（去除多餘空白、忽略大小寫）。
  - 遍歷任務列表，找到匹配且未完成的任務，將其狀態設為完成 (IsCompleted = true)。
- 回傳訊息：
  - 成功完成任務
  - 若未找到符合條件的任務
  - 若使用者無任何任務

Task reminder

```
} else if strings.HasPrefix(message.Text, "任務：") {
    // 設定作業，格式："作業：2024/12/20 12:00 統計作業 1天前"
    response := handleTaskCreation(userID, message.Text)
    if _, err := bot.ReplyMessage(event.ReplyToken, linebot.NewTextMessage(response)).Do(); err != nil {
        log.Print(err)
    }
} else if message.Text == "任務清單" {
    // 列出未完成作業
    response := listTasks(userID)
    if _, err := bot.ReplyMessage(event.ReplyToken, linebot.NewTextMessage(response)).Do(); err != nil {
        log.Print(err)
    }
} else if strings.HasPrefix(message.Text, "完成：") {
    // 標記作業完成，格式："完成：統計作業"
    taskName := strings.TrimSpace(strings.TrimPrefix(message.Text, "完成："))
    response := completeTask(userID, taskName)
    if _, err := bot.ReplyMessage(event.ReplyToken, linebot.NewTextMessage(response)).Do(); err != nil {
        log.Print(err)
    }
```

Task reminder

```go
if len(parts) < 4 {
    return "格式錯誤，請輸入「任務：YYYY/MM/DD HH:MM 作業名稱 X小時前」。"
}

deadline, err := time.Parse("2006/01/02 15:04", parts[0]+" "+parts[1])
if err != nil {
    return "日期格式錯誤，請輸入正確的格式，例如：2024/12/20 12:00。"
}

taskName := strings.Join(parts[2:len(parts)-1], " ")
reminderDuration := strings.TrimSuffix(parts[len(parts)-1], "小時前")
hoursBefore, err := strconv.Atoi(reminderDuration)
if err != nil || hoursBefore < 0 {
    return "提醒時間格式錯誤，請輸入正確的格式，例如：12小時前。"
}

// 計算提醒時間
reminderTime := deadline.Add(-time.Duration(hoursBefore) * time.Hour)
if reminderTime.Before(time.Now()) {
    return "提醒時間已過，請設定未來的提醒時間。"
}
```

```go
task := Task{
    Name:        taskName,
    Deadline:    deadline,
    Reminder:    reminderTime,
    IsCompleted: false,
}
tasks[userID] = append(tasks[userID], task)
```

```go
// Goroutine
go scheduleReminder(userID, task)
```

Task reminder

```go
func scheduleReminder(userID string, task Task) {
    duration := time.Until(task.Reminder)
    if duration <= 0 {
        log.Printf("提醒時間已過，無法設置提醒：%s", task.Name)
        return
    }

    time.Sleep(duration)

    // 發送提醒
    if !task.IsCompleted {
        message := fmt.Sprintf("提醒：任務「%s」的截止時間是：%s。請記得完成！",
            task.Name, task.Deadline.Format("2006/01/02 15:04"))
        if _, err := bot.PushMessage(userID, linebot.NewTextMessage(message)).Do(); err != nil {
            log.Print(err)
        }
    }
}
```

Task reminder

```go
var pendingTasks []Task
for _, task := range tasks {
    if !task.IsCompleted && task.Deadline.After(time.Now()) {
        pendingTasks = append(pendingTasks, task)
    }
}
```

```go
sort.Slice(pendingTasks, func(i, j int) bool {
    return pendingTasks[i].Deadline.Before(pendingTasks[j].Deadline)
})
```

```go
// output
var response strings.Builder
response.WriteString("未完成任務清單：\n")
for _, task := range pendingTasks {
    response.WriteString(fmt.Sprintf(
        "- %s\n截止時間：%s\n提醒時間：%s\n",
        task.Name, task.Deadline.Format("2006/01/02 15:04"),
        task.Reminder.Format("2006/01/02 15:04"),
    ))
}
```

Demo

Scan me

END