# ExSeedアウトプット講義

## ～設計課題～

氏名：今井 智也

＜設計図テンプレート＞

---

### 企画・目標の設定

ユーザーゴール

- RAG機能の付いたチャットボットを導入し業務効率の改善を行いたい
- 正確で信頼性の高い一貫性のある情報を提供する
- 過去のやり取りからより特化した対応を行えるようにする

ビジネスゴール

- 業務プロセスを削減することで効率化を行い運営コストの削減を行う
- データを収集しサービスの改善や戦略の向上を行う
- 業務におけるミスを低減させビジネス運営のリスクを軽減する

---

### 要件定義

機能要件

- 文字を打ち込むときちんと返答を行う
- 分からないときはわからないと返答する
- ユーザーの質問を理解し適切な回答を生成する
- 過去のやり取りからより特化した回答を行えるようにする
- よくある質問を管理して効率的に提供を行う、フィードバックの収集
- 解決できない場合の対応策
- 質問をリセットする機能
- 複数のチャンネルからの問い合わせに対応できるようにする
- 信頼性の高い情報のみを提供する

非機能要件

- 常に利用できること
- 利用頻度の高まりにきちんと対応できる　フリーズしない
- データの保護とプライバシーの確保
- 最低限のレスポンス速度

- 更新やシステムの追加を行いやすいようにする
- ユーザーインターフェース(UI)を考える

---

## 内部設計　～ユーザー視点での挙動～

機能設計

- LMM を利用して質問内容を理解し適切な回答をするものを選ぶ
  ・今回は Pinecone を利用する（Pinecone は高い検索性能の高さと実装のしやすさがある。正確性の面では Azule AI Search に劣る可能性があるがコストが高い欠点があり ChromaDB は視覚的特徴をもとにした検索に適しているため今回は不適と判断した）
- 質問場所が見てわかりやすい設計にする
  ・UI の工夫
- 応答速度が速い
  ・LMM に頑張ってもらう
- リセットボタンや新規ボタンがある
  ・機能追加
- 入力を自由にできる
- 応答できない場合の対応策

画面設計

- シンプルな UI を利用　見覚えのある形(Web 検索サイトなど)を模倣することで使い方が分かりやすいデザインにする)
- 対話型のインターフェース　(chat gpt のようなデザインがわかりやすい)
- 視覚的に今どの状態なのかわかるように表示する
- 複数の言語に対応
- エラーに対応

---

## 内部設計　～具体的にどう実装するか～

データ設計

- ユーザー情報(UserID,UserName,Preferences,Language)
- 質問履歴(QueryID,UserID,QueryText,Timestamp)
- 応答履歴(ResponceID,QueryID,ResponseText,Timestamp)
- 知識ベース(ArticleID,Title,Content,Category)

モジュール設計

・変数の設定

- user_query:質問格納

- processed_query:前処理を施した後格納
- query_intent:質問を解析した後格納
- query_entities:重要な情報を格納
- response:最終的な回答を格納する
- 質問を受け取り前処理を施す関数
- 前処理された情報を解析する関数
- 回答を生成する関数
- 回答を返信する関数
- 今までの質問を clear する関数

疑似コード

```
def process_query(user_query):
    processed_query = preprocess(user_query)
    query_intent, query_entities = analyze_query(processed_query)
    response = find_or_generate_response(query_intent, query_entities)
    return response

def preprocess(query):
    # 文字列の前処理を行う
    return processed_query

def analyze_query(processed_query):
    # モデルを使用して意図とエンティティを解析
    return query_intent, query_entities

def find_or_generate_response(intent, entities):
    # データベース検索または回答生成のロジック
    return response

def clear():
    #今までの操作を取り消す


# ユーザーからの質問を受け取る
user_query = "業務の***について知りたいです"
# 回答プロセスを実行
response = process_query(user_query)
print(response)
```

プログラム設計

- User_query を受け取り前処理を行う
- LLM で内容を解析
- Pinecone を用いて知識ベースを検索し情報を抽出

- Response の作成
- Clear の作成