Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CS7IS5 - Adaptive Applications

GRE ADAPTIVE LEARNING APPLICATION

Team Royal

Goal: An adaptive GRE test prompt application, which understands the user’s interactions and its goals to provide a much better learning experience.

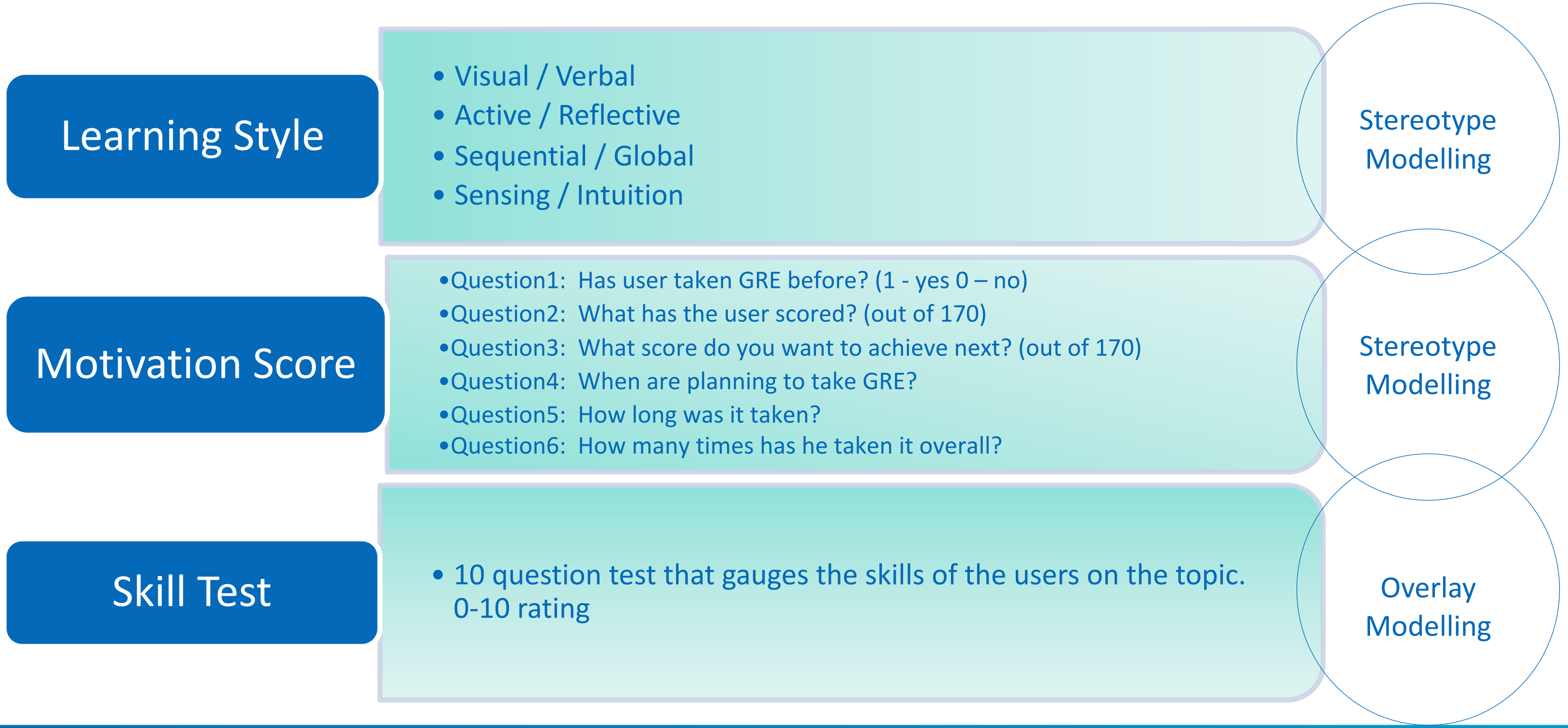
What we were able to achieve:

- A web app that provides test taking interface
- An ever expanding question database powered by open-ai’s gpt engine , with ontological tags attached to them
- A rule based engine that builds a user model
- A probabilistic model which takes user’s skill set and motivation as input parameters and outputs a “sentence” that can trigger open-ai’s gpt to query a question.

User Modelling

Blended user modelling approach (Explicit & Implicit)

- Explicit
  - Background Information
  - Learning Style – FSLSM Questionnaire
  - Motivation
- Implicit
  - User Interaction (~Motivation)
  - User Performance



Learning Style

- Visual / Verbal
- Active / Reflective
- Sequential / Global
- Sensing / Intuition

Stereotype Modelling

Motivation Score

- Question1: Has user taken GRE before? (1 - yes 0 – no)
- Question2: What has the user scored? (out of 170)
- Question3: What score do you want to achieve next? (out of 170)
- Question4: When are planning to take GRE?
- Question5: How long was it taken?
- Question6: How many times has he taken it overall?

Stereotype Modelling

Skill Test

- 10 question test that gauges the skills of the users on the topic. 0-10 rating

Overlay Modelling

User Modelling

Learning Style

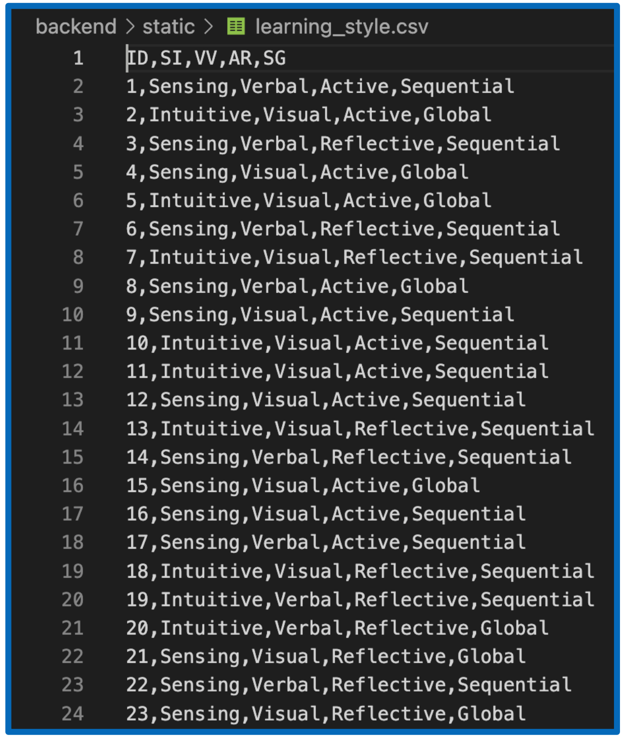
Motivation Score

Skill Test

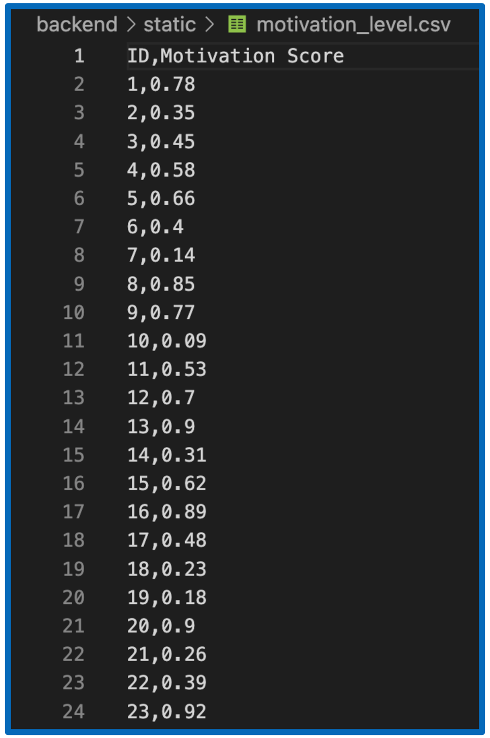
Learning_style.csv

Motivation_level.csv

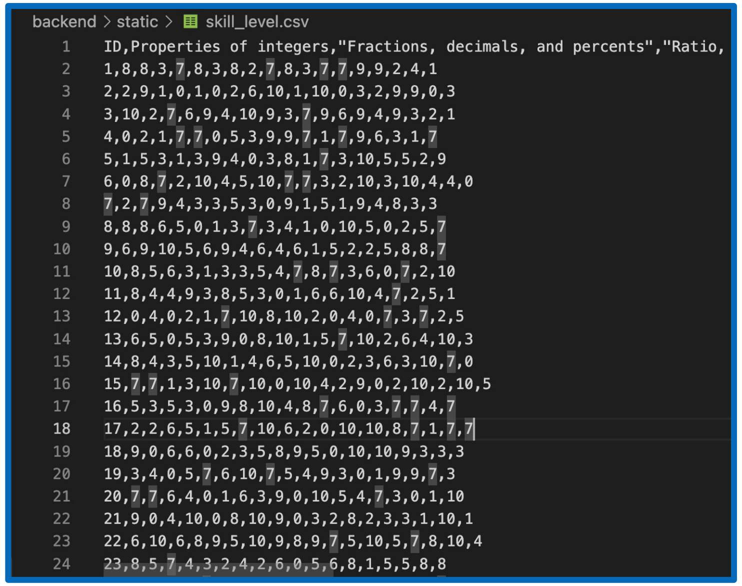
Skill_level.csv


backend > static > learning_style.csv

```
ID,SI,VV,AR,SG
1,Sensing,Verbal,Active,Sequential
2,Intuitive,Visual,Active,Global
3,Sensing,Verbal,Reflective,Sequential
4,Sensing,Visual,Active,Global
5,Intuitive,Visual,Active,Global
6,Sensing,Verbal,Reflective,Sequential
7,Intuitive,Visual,Reflective,Sequential
8,Sensing,Verbal,Active,Global
9,Sensing,Visual,Active,Sequential
10,Intuitive,Visual,Active,Sequential
11,Intuitive,Visual,Active,Sequential
12,Sensing,Visual,Active,Sequential
13,Intuitive,Visual,Reflective,Sequential
14,Sensing,Verbal,Reflective,Sequential
15,Sensing,Visual,Active,Global
16,Sensing,Visual,Active,Sequential
17,Sensing,Verbal,Active,Sequential
18,Intuitive,Visual,Reflective,Sequential
19,Intuitive,Verbal,Reflective,Sequential
20,Intuitive,Verbal,Reflective,Global
21,Sensing,Visual,Reflective,Global
22,Sensing,Verbal,Reflective,Sequential
23,Sensing,Visual,Reflective,Global
```

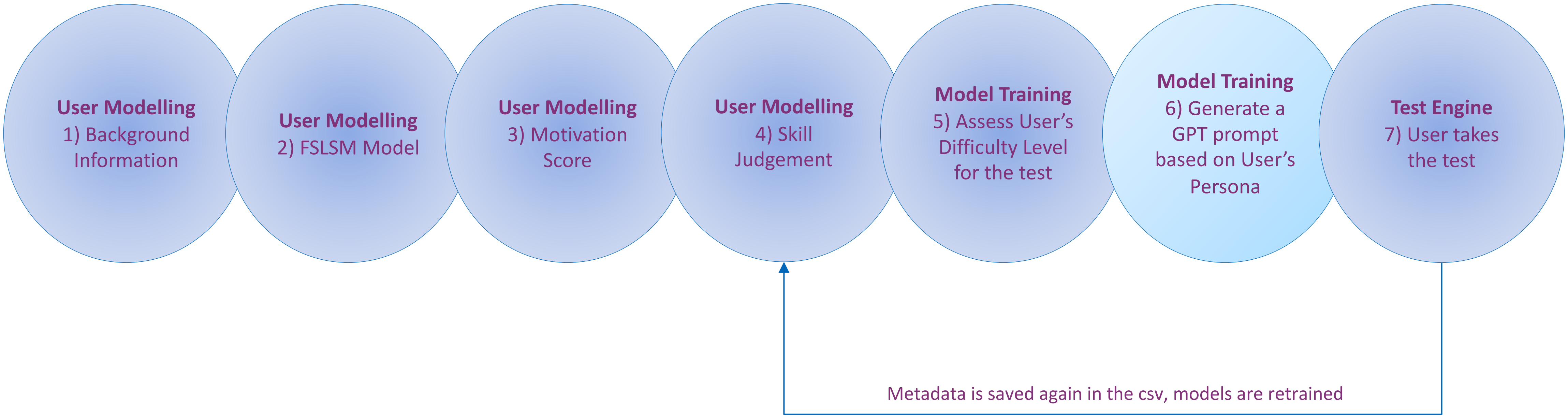

backend > static > motivation_level.csv

```
ID,Motivation Score
1,0.78
2,0.35
3,0.45
4,0.58
5,0.66
6,0.4
7,0.14
8,0.85
9,0.77
10,0.09
11,0.53
12,0.7
13,0.9
14,0.31
15,0.62
16,0.89
17,0.48
18,0.23
19,0.18
20,0.9
21,0.26
22,0.39
23,0.92
```


backend > static > skill_level.csv

```
ID,Properties of integers,"Fractions, decimals, and percents","Ratio,
1,8,8,3,7,8,3,8,2,7,8,3,7,7,9,9,2,4,1
2,2,9,1,0,1,0,2,6,10,1,10,0,3,2,9,9,0,3
3,10,2,7,6,9,4,10,9,3,7,9,6,9,4,9,3,2,1
4,0,2,1,7,7,0,5,3,9,9,7,1,7,9,6,3,1,7
5,1,5,3,1,3,9,4,0,3,8,1,7,3,10,5,5,2,9
6,0,8,7,2,10,4,5,10,7,7,3,2,10,3,10,4,4,0
7,2,7,9,4,3,3,5,3,0,9,1,5,1,9,4,8,3,3
8,8,8,6,5,0,1,3,7,3,4,1,0,10,5,0,2,5,7
9,6,9,10,5,6,9,4,6,4,6,1,5,2,2,5,8,8,7
10,8,5,6,3,1,3,3,5,4,7,8,7,3,6,0,7,2,10
11,8,4,4,9,3,8,5,3,0,1,6,6,10,4,7,2,5,1
12,0,4,0,2,1,7,10,8,10,2,0,4,0,7,3,7,2,5
13,6,5,0,5,3,9,0,8,10,1,5,7,10,2,6,4,10,3
14,8,4,3,5,10,1,4,6,5,10,0,2,3,6,3,10,7,0
15,7,7,1,3,10,7,10,0,10,4,2,9,0,2,10,2,10,5
16,5,3,5,3,0,9,8,10,4,8,7,6,0,3,7,7,4,7
17,2,2,6,5,1,5,7,10,6,2,0,10,10,8,7,1,7,7
18,9,0,6,6,0,2,3,5,8,9,5,0,10,10,9,3,3,3
19,3,4,0,5,7,6,10,7,5,4,9,3,0,1,9,9,7,3
20,7,7,6,4,0,1,6,3,9,0,10,5,4,7,3,0,1,10
21,9,0,4,10,0,8,10,9,0,3,2,8,2,3,3,1,10,1
22,6,10,6,8,9,5,10,9,8,9,7,5,10,5,7,8,10,4
23,8,5,7,4,3,2,4,2,6,0,5,6,8,1,5,5,8,8
```

How it works



User Modelling
1) Background Information

User Modelling
2) FSLSM Model

User Modelling
3) Motivation Score

User Modelling
4) Skill Judgement

Model Training
5) Assess User’s Difficulty Level for the test

Model Training
6) Generate a GPT prompt based on User’s Persona

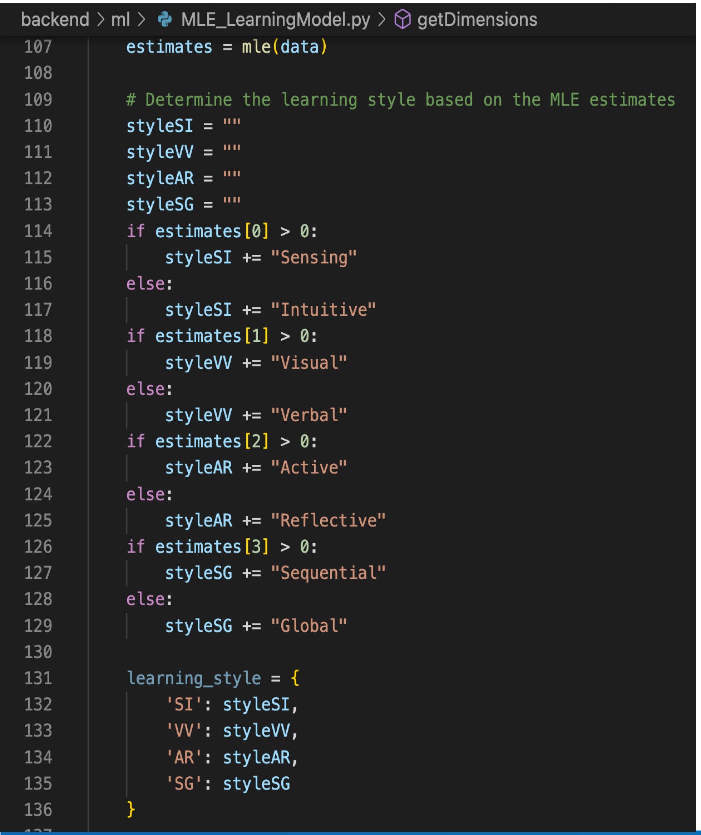
Test Engine
7) User takes the test

Metadata is saved again in the csv, models are retrained

Clean Code

Frontend - Test Interface (React + Next.js)
Middleware - GraphQL APIs
Backend - Test Engine , User Assessment Engine (Python)

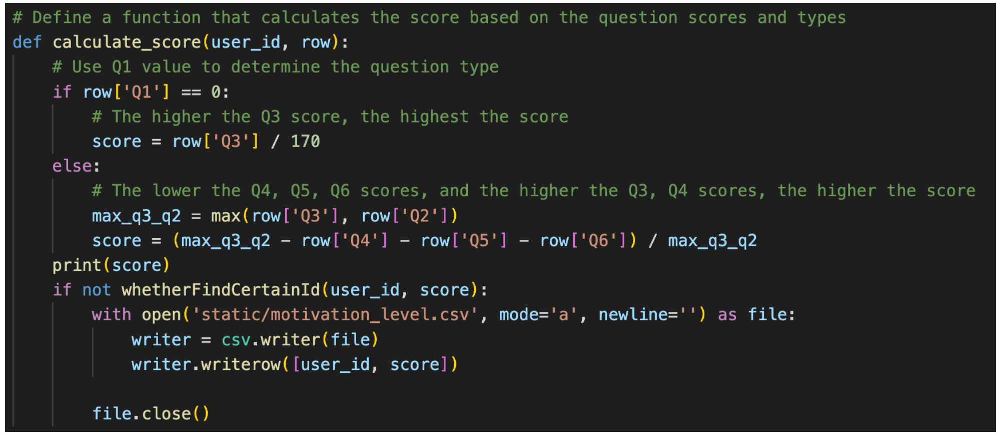
1) Learning Styles for User



backend > ml > MLE_LearningModel.py > getDimensions

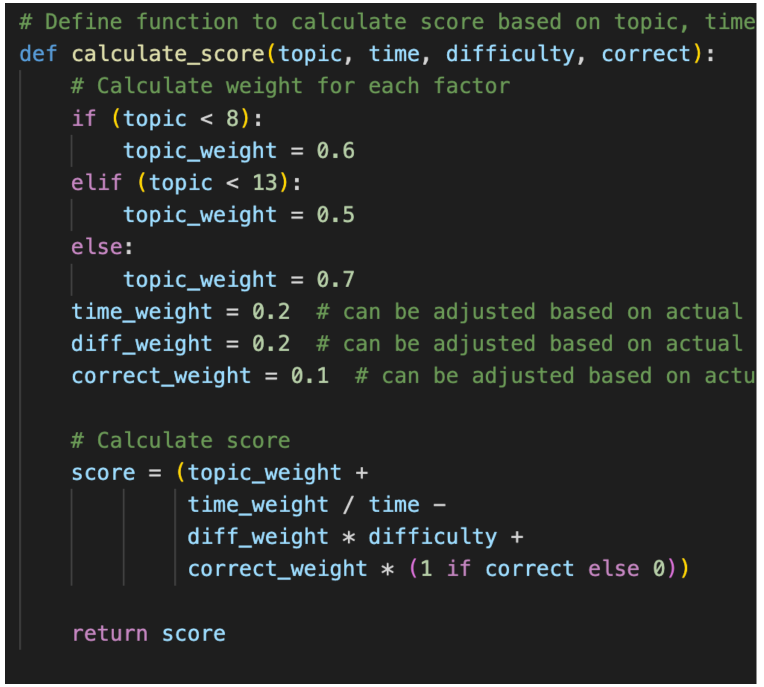
```python
    estimates = mle(data)

    # Determine the learning style based on the MLE estimates
    styleSI = ""
    styleVV = ""
    styleAR = ""
    styleSG = ""
    if estimates[0] > 0:
        styleSI += "Sensing"
    else:
        styleSI += "Intuitive"
    if estimates[1] > 0:
        styleVV += "Visual"
    else:
        styleVV += "Verbal"
    if estimates[2] > 0:
        styleAR += "Active"
    else:
        styleAR += "Reflective"
    if estimates[3] > 0:
        styleSG += "Sequential"
    else:
        styleSG += "Global"

    learning_style = {
        'SI': styleSI,
        'VV': styleVV,
        'AR': styleAR,
        'SG': styleSG
    }
```

2) Calculates Motivation score based on a simple formula



```python
# Define a function that calculates the score based on the question scores and types
def calculate_score(user_id, row):
    # Use Q1 value to determine the question type
    if row['Q1'] == 0:
        # The higher the Q3 score, the highest the score
        score = row['Q3'] / 170
    else:
        # The lower the Q4, Q5, Q6 scores, and the higher the Q3, Q4 scores, the higher the score
        max_q3_q2 = max(row['Q3'], row['Q2'])
        score = (max_q3_q2 - row['Q4'] - row['Q5'] - row['Q6']) / max_q3_q2
    print(score)
    if not whetherFindCertainId(user_id, score):
        with open('static/motivation_level.csv', mode='a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow([user_id, score])

        file.close()
```

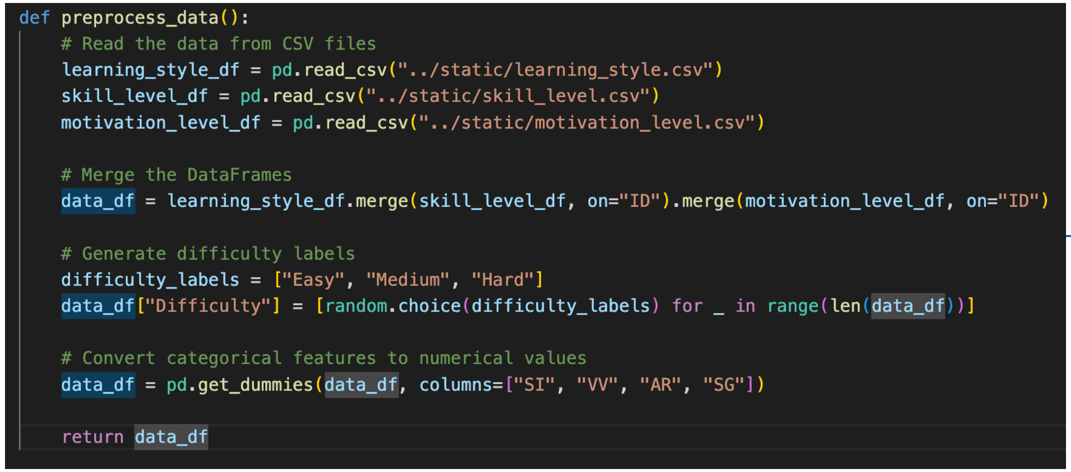
3) Calculates score for each topic based on test parameters



```python
# Define function to calculate score based on topic, time
def calculate_score(topic, time, difficulty, correct):
    # Calculate weight for each factor
    if (topic < 8):
        topic_weight = 0.6
    elif (topic < 13):
        topic_weight = 0.5
    else:
        topic_weight = 0.7
    time_weight = 0.2  # can be adjusted based on actual
    diff_weight = 0.2  # can be adjusted based on actual
    correct_weight = 0.1  # can be adjusted based on actu

    # Calculate score
    score = (topic_weight +
             time_weight / time -
             diff_weight * difficulty +
             correct_weight * (1 if correct else 0))

    return score
```

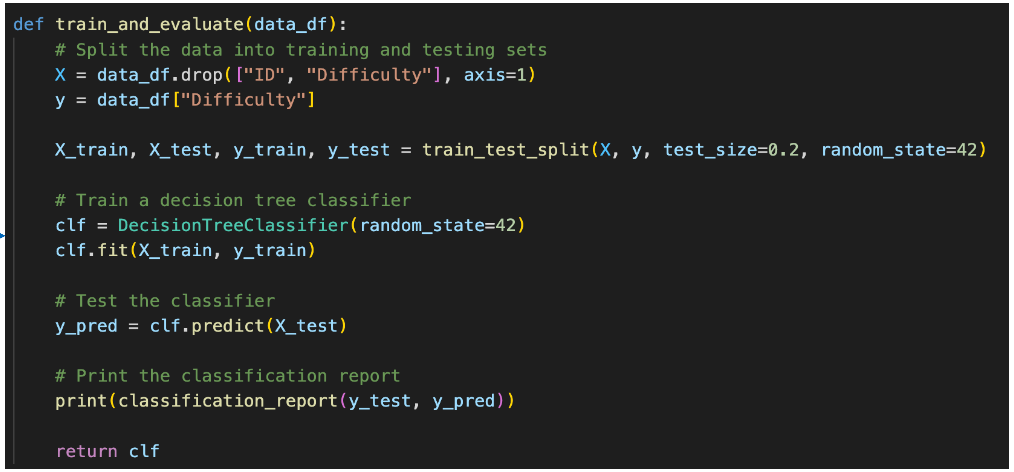
Clean Code

Frontend - Test Interface (React + Next.js)
Middleware - GraphQL APIs
Backend - Test Engine , User Assessment Engine (Python)



```python
def preprocess_data():
    # Read the data from CSV files
    learning_style_df = pd.read_csv("../static/learning_style.csv")
    skill_level_df = pd.read_csv("../static/skill_level.csv")
    motivation_level_df = pd.read_csv("../static/motivation_level.csv")

    # Merge the DataFrames
    data_df = learning_style_df.merge(skill_level_df, on="ID").merge(motivation_level_df, on="ID")

    # Generate difficulty labels
    difficulty_labels = ["Easy", "Medium", "Hard"]
    data_df["Difficulty"] = [random.choice(difficulty_labels) for _ in range(len(data_df))]

    # Convert categorical features to numerical values
    data_df = pd.get_dummies(data_df, columns=["SI", "VV", "AR", "SG"])

    return data_df
```

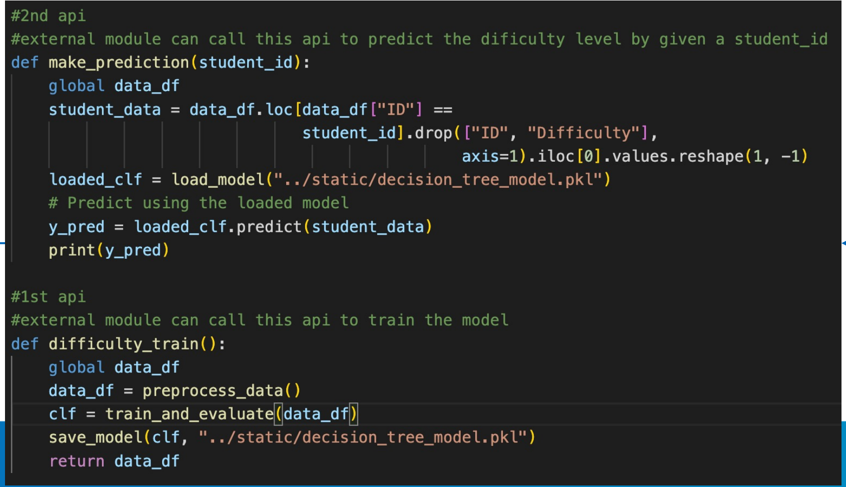
1) Preprocess the data



```python
def train_and_evaluate(data_df):
    # Split the data into training and testing sets
    X = data_df.drop(["ID", "Difficulty"], axis=1)
    y = data_df["Difficulty"]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Train a decision tree classifier
    clf = DecisionTreeClassifier(random_state=42)
    clf.fit(X_train, y_train)

    # Test the classifier
    y_pred = clf.predict(X_test)

    # Print the classification report
    print(classification_report(y_test, y_pred))

    return clf
```

2) Train and evaluate

3) Use the mode to make predcitions

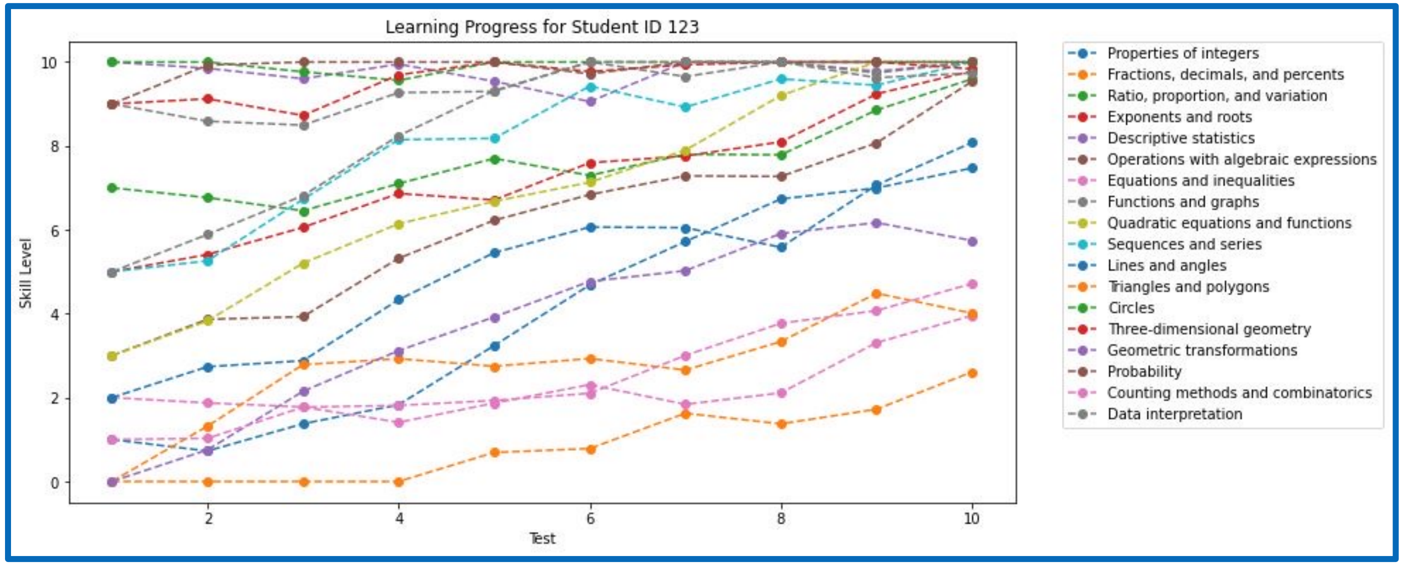


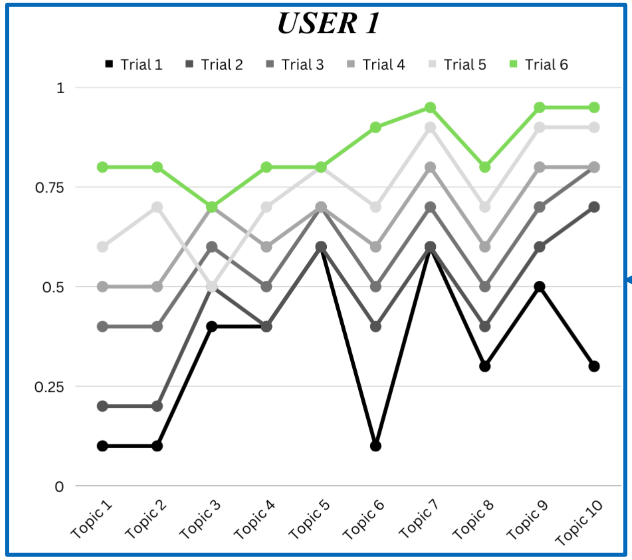
```python
#2nd api
#external module can call this api to predict the dificulty level by given a student_id
def make_prediction(student_id):
    global data_df
    student_data = data_df.loc[data_df["ID"] ==
                               student_id].drop(["ID", "Difficulty"],
                                                axis=1).iloc[0].values.reshape(1, -1)
    loaded_clf = load_model("../static/decision_tree_model.pkl")
    # Predict using the loaded model
    y_pred = loaded_clf.predict(student_data)
    print(y_pred)

#1st api
#external module can call this api to train the model
def difficulty_train():
    global data_df
    data_df = preprocess_data()
    clf = train_and_evaluate(data_df)
    save_model(clf, "../static/decision_tree_model.pkl")
    return data_df
```

4) User Interaction

In Action?

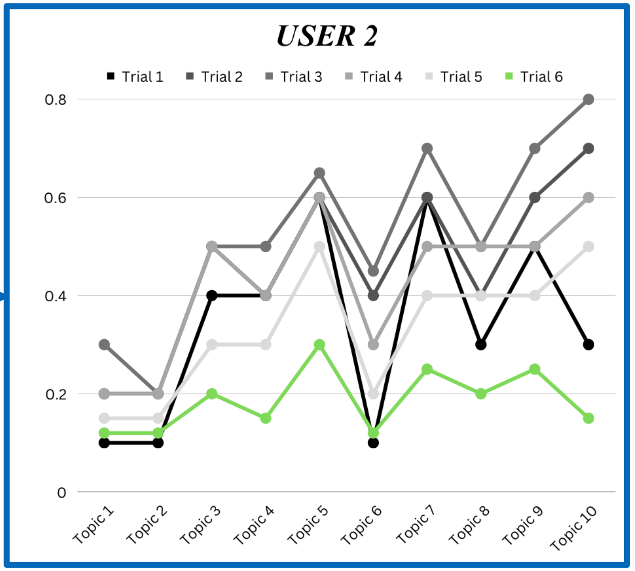
Interpretation of Data



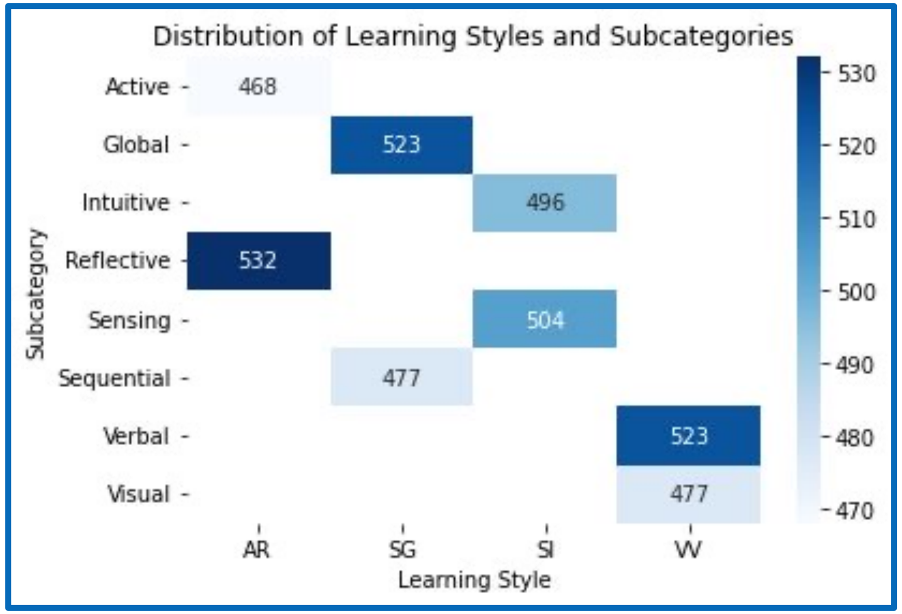


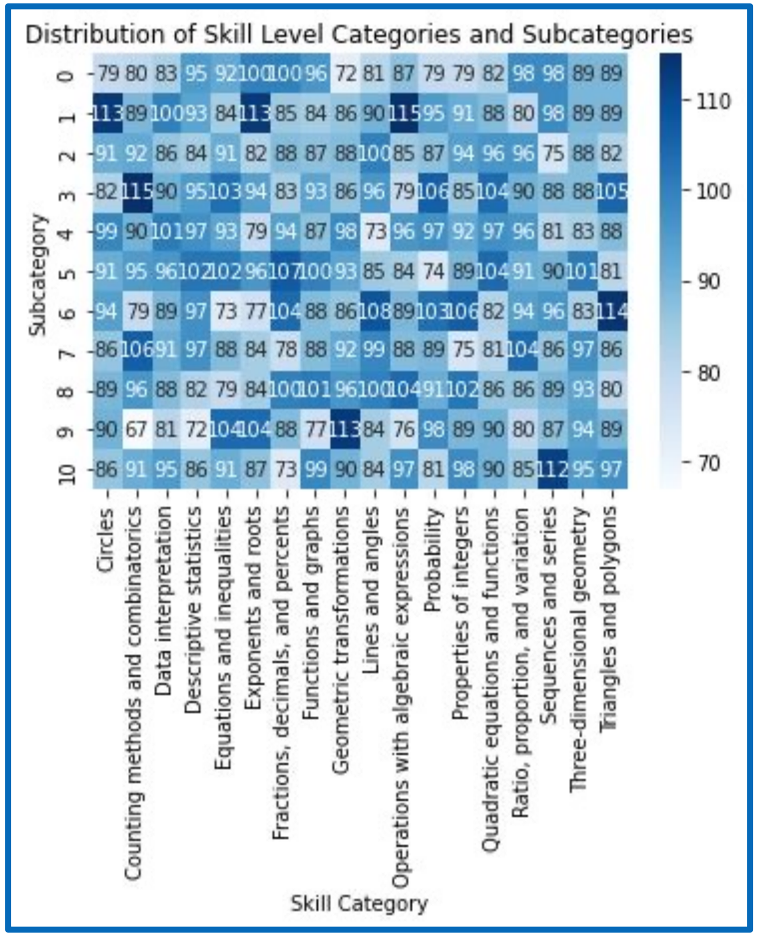
1) User is motivated and practices everyday

2) User is motivated and doesn’t practice everyday



Interpretation of Data





Thank you

Siyu Liao - 22323209 - liaosi@tcd.ie
Haoxian Liu - 22322820 - liuha@tcd.ie
Rui Zhao - 22328549 - zhaoru@tcd.ie
Pallavit Aggarwal - 22333721 - aggarwpa@tcd.ie
Shritesh Jamulkar - 22324542 - jamulks@tcd.ie