TECHNE

Member 1: Nikita
Member 2: Shailendra Pratap Singh
Member 3: Subrat Sahil Gupta

Problem Statement

FIT SCORE: NON-INVASIVE HEALTH MONITORING AND DIAGNOSTICS

Solution:

- The fitness score app utilizes a Long Short Term Memory (LSTM) model to analyze users' health vitals and fitness parameters, enabling it to classify fitness scores for each individual users and providing deeply analysed prompt.
- The app not only predicts personalized fitness scores but also provides detailed information about each health vital.
- The color-coded visuals allow users to quickly understand the state of their health vitals and take necessary actions to address potential health issues proactively.
- Using Overall Fitness Score, it plots a line graph showcasing fitness score trends over time, and generates tailored prompts; empowering users to make proactive decisions to improve their well-being.

Model

Pre-Processing

```
# Loaing dataset in dataframe 'df0'
df0 = pd.read_csv('original_data.csv')

# Assuming 'Gender' is a categorical variable that needs encoding
df0['Gender'] = df0['Gender'].map({'Male': 0, 'Female': 1})

# Preprocess the 'Blood Pressure' column
df0[['Systolic', 'Diastolic']] = df0['Blood Pressure'].str.split('/', expand=True)
df0[['Systolic', 'Diastolic']] = df0[['Systolic', 'Diastolic']].astype(float)

# Drop the original 'Blood Pressure' column
df0.drop(columns=['Blood Pressure'], inplace=True)
```

Contd...

```
weights = {
    'Gender': 0.05,
    'Age': 0.1,
    'Sleep Duration': 0.2,
    'Stress Level': 0.15,
    'Heart Rate': 0.15,
    'Daily Steps': 0.1,
    'Systolic': 0.15,
    'Diastolic': 0.1
}

col= ['Gender','Age', 'Sleep Duration', 'Stress Level', 'Heart Rate', 'Daily Steps', 'Systolic', 'Diastolic']

# Calculate the fitness score using the defined weights
df0['Fitness Score'] = df0[col].dot(pd.Series(weights))
```

```
# Save the updated dataset with the fitness score
df0.to_csv('hv.csv', index=False)
```

Splitting and Normalization

```
# Load the dataset
df = pd.read_csv('hv.csv')

# Separate features and target variable
X = df.drop(columns=['Fitness Score'])
y = df['Fitness Score']



# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=28)
```

```
# Normalize the features
scaler_X = MinMaxScaler()
X_train_scaled = scaler_X.fit_transform(X_train)
X_test_scaled = scaler_X.transform(X_test)

# Normalize the target
scaler_y = MinMaxScaler()
y_train_scaled = scaler_y.fit_transform(y_train.values.reshape(-1, 1))
```

Model Structure & Prediction

```
# Build the Neural Network model
model = Sequential()
model.add(Dense(64, input_dim=X_train_scaled.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='linear'))


# Compile the model
optimizer = Nadam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='mse')


# Train the model
history = model.fit(X_train_scaled, y_train_scaled, epochs=180, batch_size=32, validation_split=0.2)
```

```
# Make predictions
X_test_scaled = scaler_X.transform(X_test)
y_pred_scaled = model.predict(X_test_scaled).flatten()

# Inverse transform to get values in the original range
y_pred = scaler_y.inverse_transform(y_pred_scaled.reshape(-1, 1)).flatten()
```

Performance Evaluation & Saving

```
# Calculate errors
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)

print(f'Mean Squared Error: {mse}')
print(f'R-squared (R2) Score: {r2}')
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
```

```
Mean Squared Error: 10.194154458954516
R-squared (R2) Score: 0.9996057267670587
Mean Absolute Error (MAE): 1.7597557617187416
Root Mean Squared Error (RMSE): 3.192828598430319
```

```
# Save the trained model to an HDF5 file
model.save('techne_ANN.h5')

# Save the scalers to pickle files
with open('scaler_X.pkl', 'wb') as f:
    pickle.dump(scaler_X, f)

with open('scaler_y.pkl', 'wb') as f:
    pickle.dump(scaler_y, f)
```

App Description

- Our health tracker app combines React and machine learning to help you monitor your fitness.
- Tracking sleep duration, heart rate, daily steps, blood pressure, and stress levels; this model calculates your personalized fitness score, offering insights into your health and progress.
- But it doesn't stop there! The app gives helpful advice based on your over-all fitness score trends. Stay motivated with tailored prompts.
- It encourages you to stay consistent and reach your fitness goals.With user-friendly design and powerful analytics, our app empowers you to make informed decisions for a healthier and happier life.

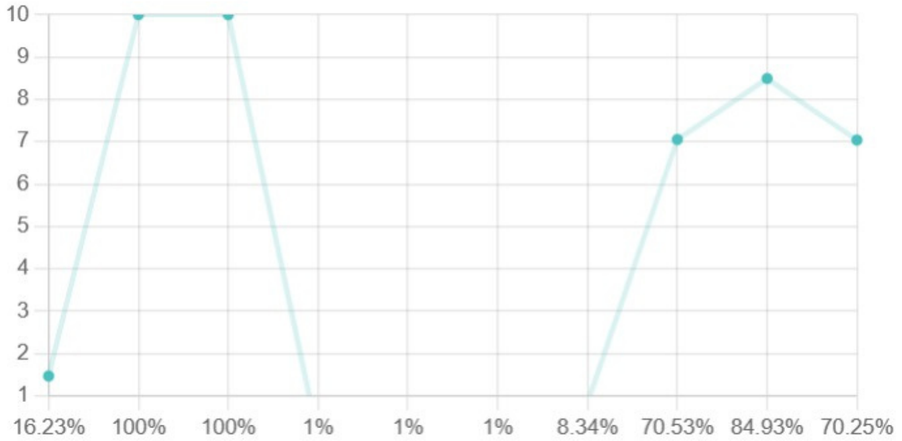
Backend Connectivity

- Train and Save the Machine Learning Model.
- Set up a Python backend using a web framework like Flask. This backend will act as the server that receives requests, processes data, and sends responses.
- Load the Model in the Backend.
- Create API endpoints in the backend to handle incoming requests related to the machine learning model. These endpoints will accept data needed for predictions, call the loaded model to make predictions, and return the results as responses.
- In the backend, use request handlers to receive and process incoming data from the client. After obtaining the required data, pass it to the machine learning model for prediction. Then, format the prediction results into an appropriate response and send it back to the client.
- Deploy the Python backend on a server or cloud platform so that it can be accessed by the client-side application or other services.

Over-all Fitness Score

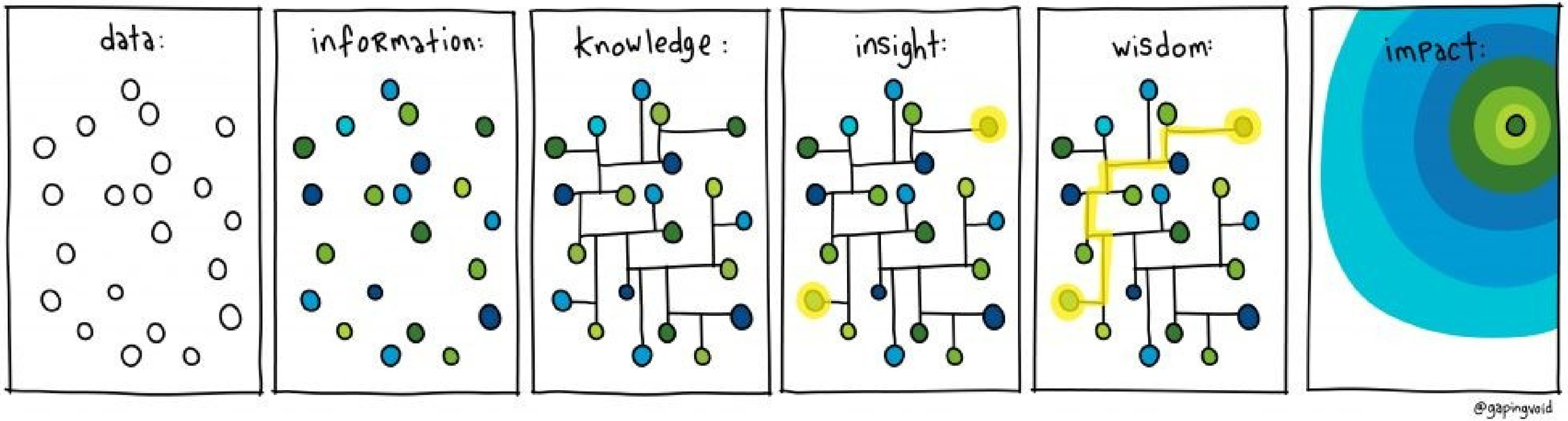
- For Overall Fitness Score, the fitness score trend based on health metrics, is graphed, and its slope indicates the rate of health improvement or decline over time.
- Prompt messages are generated, offering personalized feedback and recommendations to users.
- By incorporating machine learning and data visualization, the app provides accurate predictions and easy-to-interpret health patterns.
- The personalized prompts generate alert messages which lead to positive empowering users to make informed decisions for a healthier lifestyle.

Health Prediction App



things aren't good, Get a routine checkup

@Techne

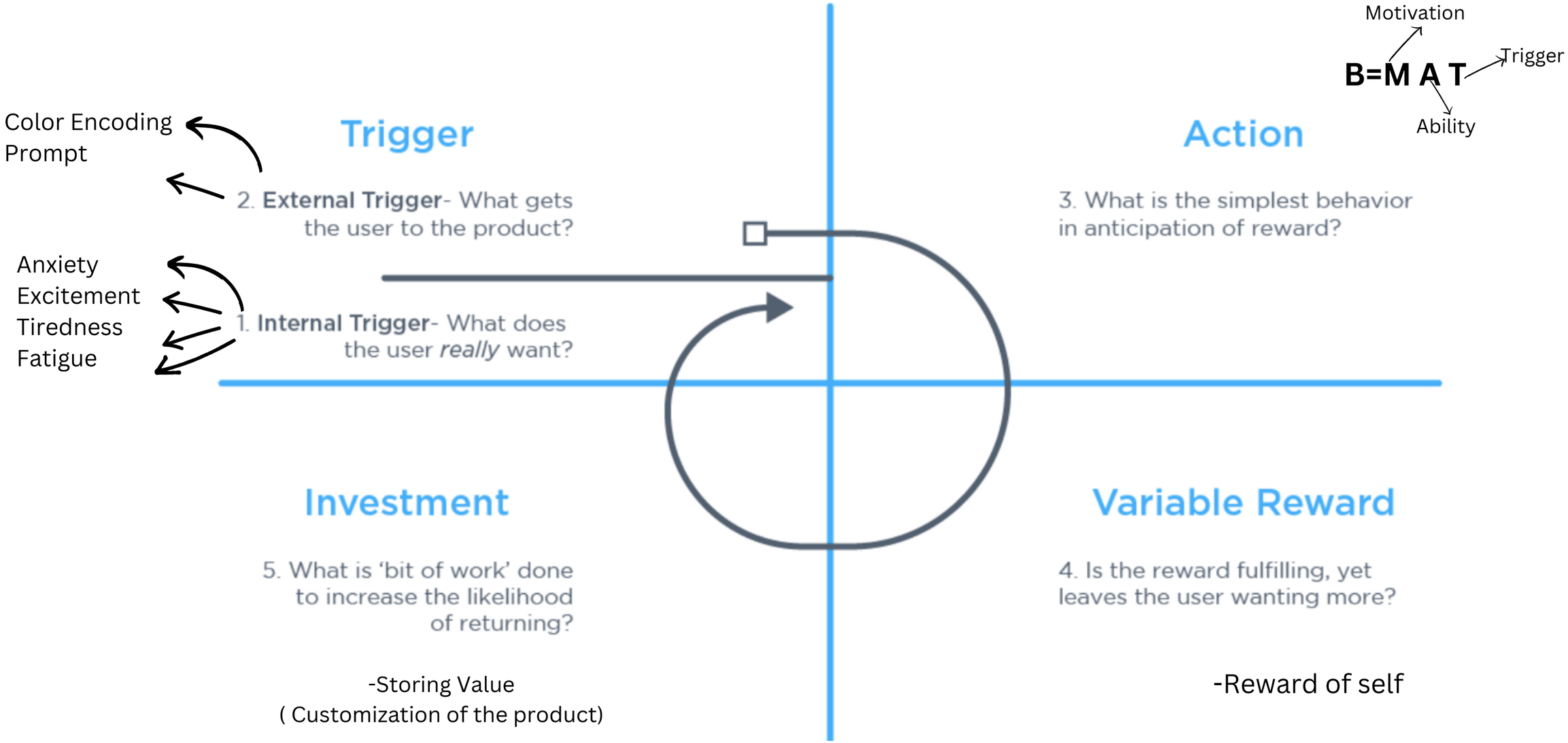

data:
information:
knowledge :
insight:
wisdom:
impact:
@gapingvoid

Habit Forming Product:

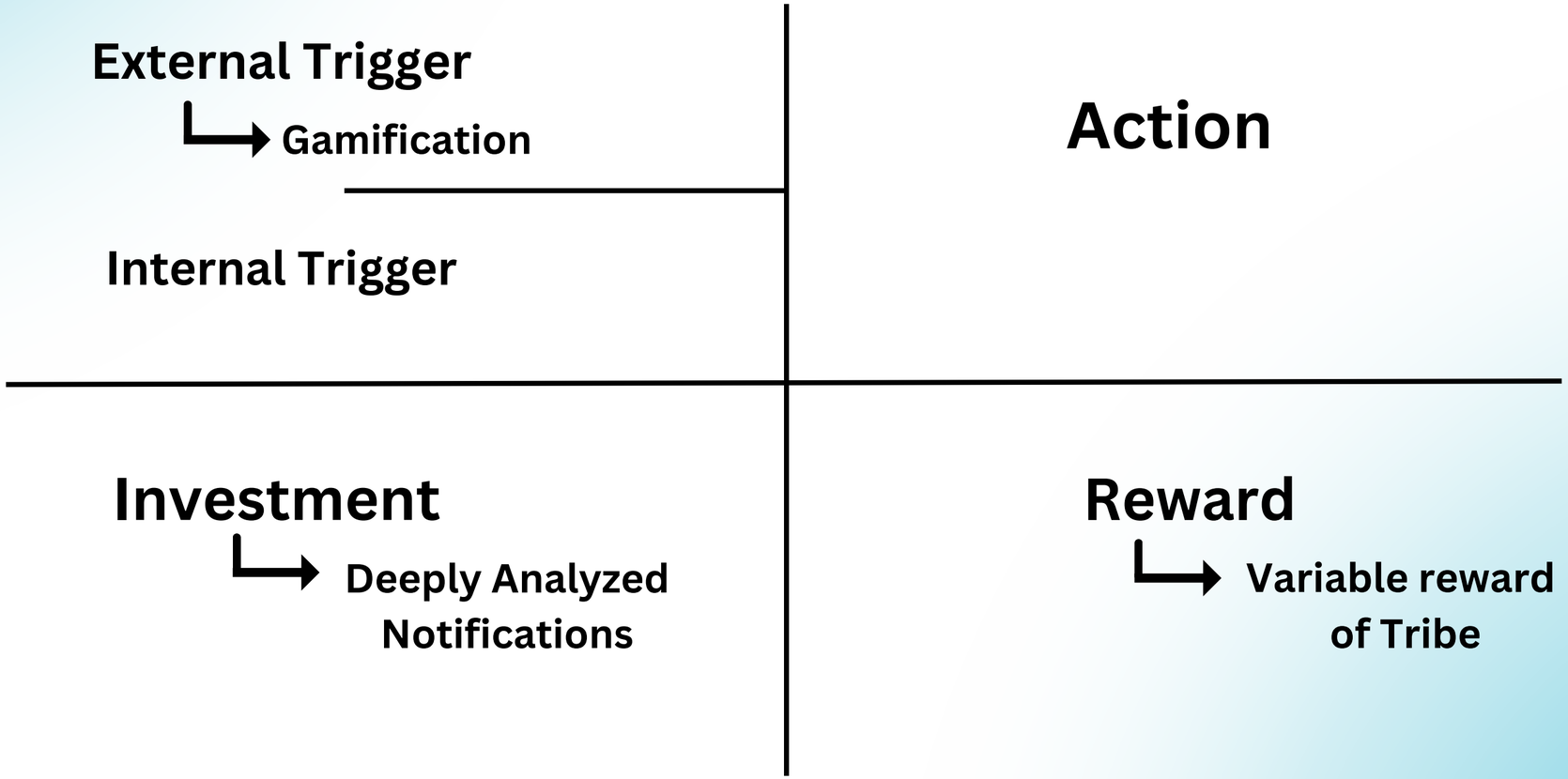
"Product should capture the monopoly of mind"

- Nir Eyal

Hook Canvas


Motivation
B=M A T
Trigger
Ability
Color Encoding
Prompt
Trigger
2. External Trigger- What gets the user to the product?
Anxiety
Excitement
Tiredness
Fatigue
1. Internal Trigger- What does the user really want?
Action
3. What is the simplest behavior in anticipation of reward?
Investment
5. What is 'bit of work' done to increase the likelihood of returning?
-Storing Value
(Customization of the product)
Variable Reward
4. Is the reward fulfilling, yet leaves the user wanting more?
-Reward of self

Future Modifications



Challenges:

Model and App connectivity - We faced challenges with Axios, a library for fetching API data. Understanding its usage, handling async functions, CORS, error management, parsing JSON, loading states, compatibility, security, testing, and API configuration were key hurdles encountered.

.

Fitness Score Prediction - The initial fitness score predicted was in scaled normalize form, similar to all the health vitals and was not providing interpretable insights for analysis.

Conclusion

- Our Fitness Score Prediction app is a meticulously designed habit-forming product, leveraging the Hook Model to empower users in taking control of their health.

- With engaging triggers, actions, rewards, and investments, it fosters positive habits and meaningful interactions; while personalized insights and wisdom guide them towards healthier choices.

- Our app makes a tangible impact, transforming lives and making users healthier and happier.

Q&A Session

Thank you

for hearing us out