In [ ]:
```
#Problem Statement

#To build a machine learning model that can predict the Stress Level of a develc
```

In [29]:
```
# ------------------------------------------------------------
# AI-Driven Prediction of Developer Stress Levels
# Full End-to-End Machine Learning Project
# ------------------------------------------------------------

# 1. Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression


from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

In [2]:
```
#load dataset csv file
df=pd.read_csv("AI_Developer_Performance_Extended_1000.csv")
df
```

Out[2]:

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cog
0	7	416	9	7	6	5.9	
1	4	269	16	13	5	5.1	
2	11	439	3	0	2	6.2	
3	8	472	15	9	4	4.2	
4	5	265	19	16	5	8.1	
...	...	...	...	...	...	...	
995	10	660	14	13	0	5.5	
996	9	484	13	11	1	8.8	
997	2	128	4	0	3	8.7	
998	8	266	5	3	1	5.7	
999	8	604	16	15	3	6.9	

1000 rows × 13 columns

In [23]:
```
df.head()  # first 5 rows and columns in dataset
```

Out[23]:

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cogni
0	7	416	9	7	6	5.9	
1	4	269	16	13	5	5.1	
2	11	439	3	0	2	6.2	
3	8	472	15	9	4	4.2	
4	5	265	19	16	5	8.1	

In [24]:
```
df.tail()     # last 5 rows and columns in dataset
```

Out[24]:

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours	Cog
995	10	660	14	13	0	5.5	
996	9	484	13	11	1	8.8	
997	2	128	4	0	3	8.7	
998	8	266	5	3	1	5.7	
999	8	604	16	15	3	6.9	

In [28]:
```
df.shape    # Shape of dataset
```

Out[28]:
```
(1000, 13)
```

In [3]:
```
df.info()   # infromation of dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Hours_Coding         1000 non-null   int64  
 1   Lines_of_Code        1000 non-null   int64  
 2   Bugs_Found           1000 non-null   int64  
 3   Bugs_Fixed           1000 non-null   int64  
 4   AI_Usage_Hours       1000 non-null   int64  
 5   Sleep_Hours          1000 non-null   float64
 6   Cognitive_Load       1000 non-null   int64  
 7   Task_Success_Rate    1000 non-null   int64  
 8   Coffee_Intake        1000 non-null   int64  
 9   Stress_Level         1000 non-null   int64  
 10  Task_Duration_Hours  1000 non-null   float64
 11  Commits              1000 non-null   int64  
 12  Errors               1000 non-null   int64  
dtypes: float64(2), int64(11)
memory usage: 101.7 KB
```

In [4]:
```
df.isna().sum()  #Check missing values
```

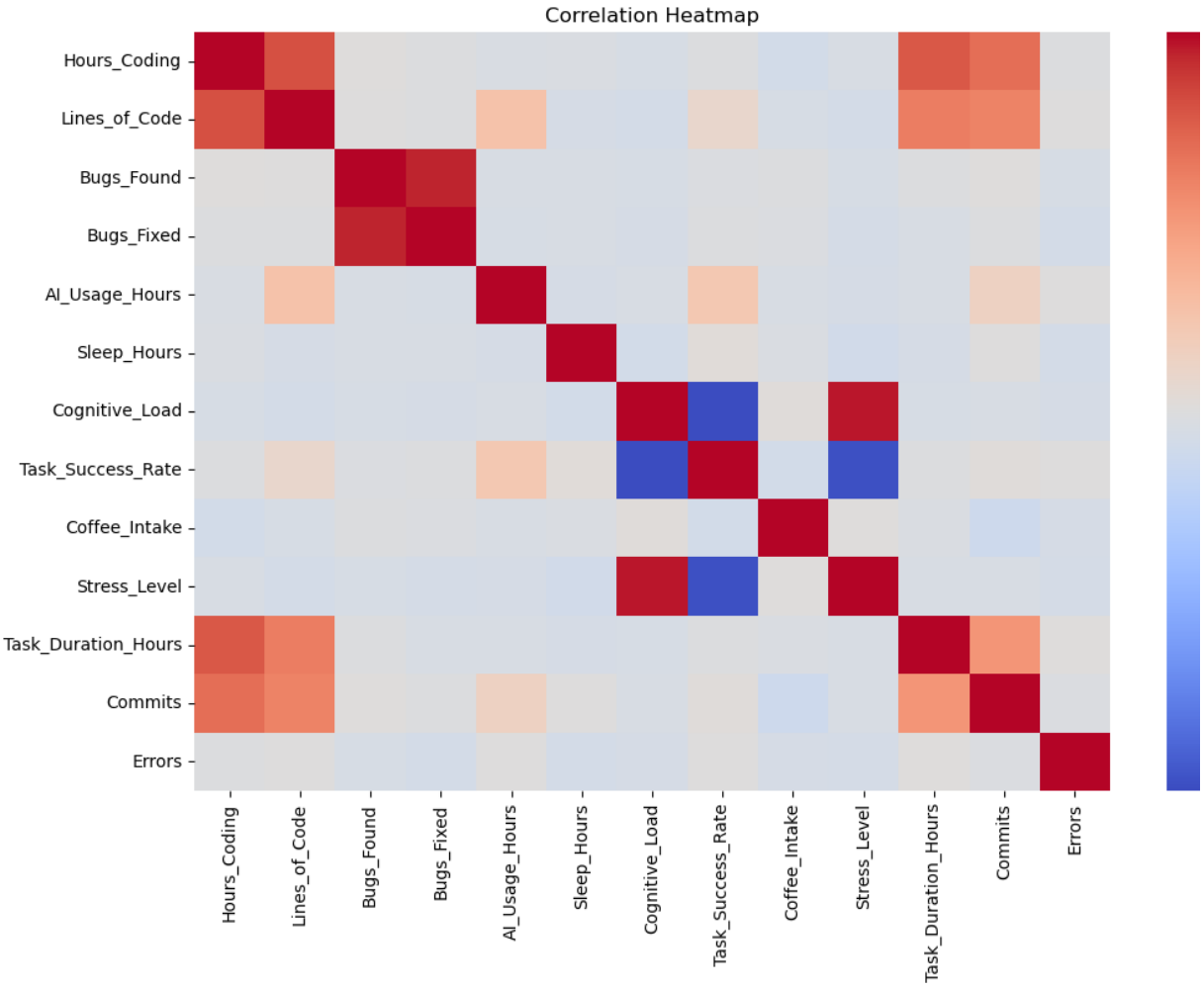
Out[4]:
```
Hours_Coding           0
Lines_of_Code          0
Bugs_Found             0
Bugs_Fixed             0
AI_Usage_Hours         0
Sleep_Hours            0
Cognitive_Load         0
Task_Success_Rate      0
Coffee_Intake          0
Stress_Level           0
Task_Duration_Hours    0
Commits                0
Errors                 0
dtype: int64
```

In [5]:
```
df.describe()       # Summary statistics
```

Out[5]:

	Hours_Coding	Lines_of_Code	Bugs_Found	Bugs_Fixed	AI_Usage_Hours	Sleep_Hours
count	1000.00000	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000
mean	5.84000	356.23400	9.876000	7.153000	2.961000	6.465800
std	3.15854	188.15535	5.796052	5.468226	2.021278	1.439529
min	1.00000	26.00000	0.000000	0.000000	0.000000	4.000000
25%	3.00000	209.50000	5.000000	2.000000	1.000000	5.200000
50%	6.00000	332.00000	10.000000	7.000000	3.000000	6.400000
75%	9.00000	480.50000	15.000000	12.000000	5.000000	7.700000
max	11.00000	993.00000	19.000000	19.000000	6.000000	9.000000

In [6]:
```
#Correlation Heatmap
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=False, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



In [8]:
```
#Feature Selection
target = "Stress_Level"
X = df.drop(columns=[target])
y = df[target]
```

In [30]:
```
#Train the Linear Regression Model
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
```

Out[30]:
```
▾ LinearRegression

LinearRegression()
```

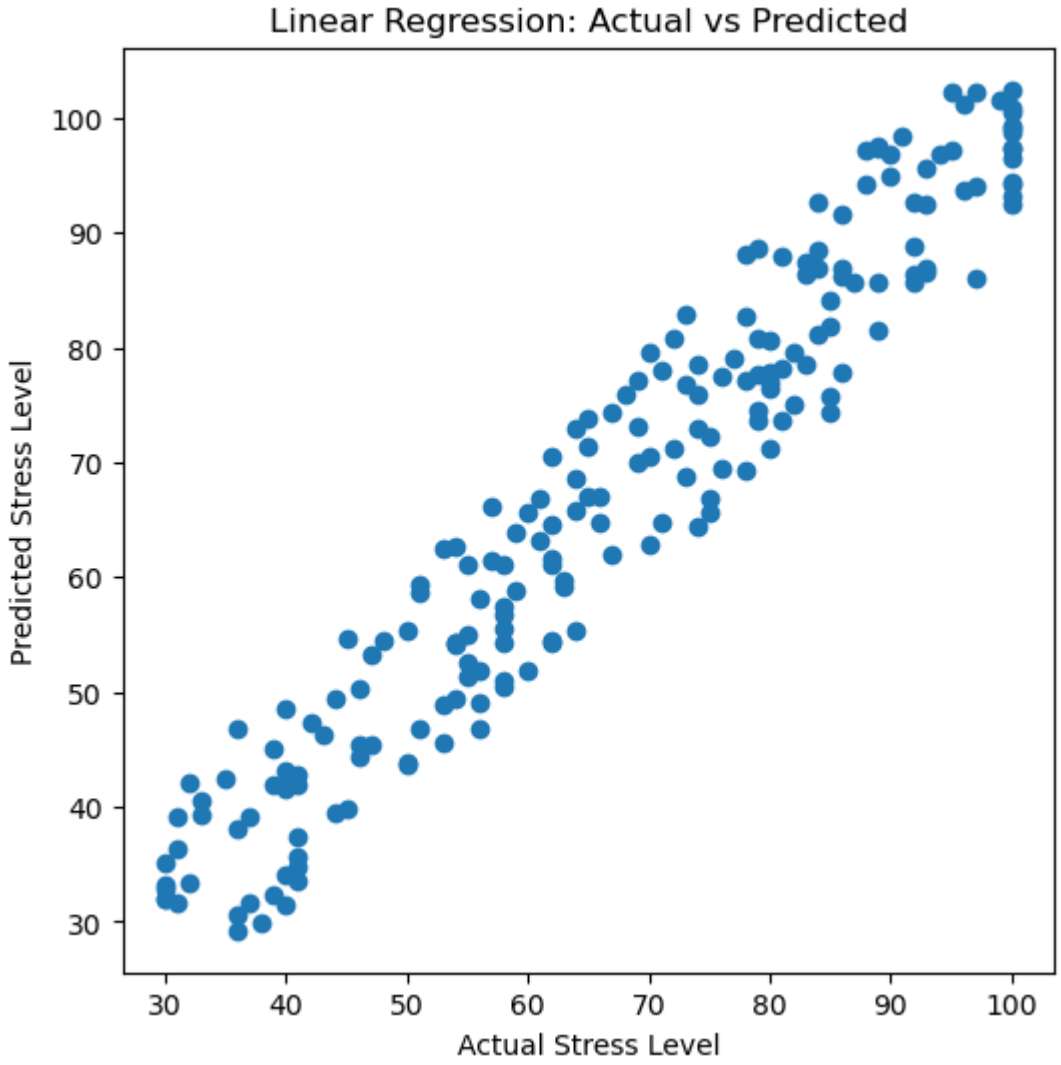
In [36]:
```
y_pred_lr = lr.predict(X_test_scaled)        # Predictions of data
y_pred_lr
```

Out[36]:
```
array([ 73.60057379,  94.33626663,  92.7310387 ,  69.50929704,
        51.31661121,  98.31461025,  61.08672148,  54.18353447,
        61.68369649,  93.16066811,  62.72283553,  65.67394796,
        55.43712123,  42.82860598,  67.05604298,  97.19858599,
        73.6323938 ,  46.86063094,  39.11070479,  29.25947897,
        58.60459104,  85.69915327,  70.52470316,  72.95700141,
        41.66195067,  72.31503438,  35.65455036,  86.38516766,
        55.47894003,  86.93607877,  85.64428093,  45.34239318,
        70.06604775,  33.47710995,  34.05970979,  61.45821736,
        42.41842621,  42.03152478,  75.91953507,  74.98708863,
        94.03642987,  49.39512375,  34.87036478,  74.28280941,
        39.43751902,  54.31462385,  99.05225369,  74.31769054,
        77.21053481,  56.73944612,  76.46852394,  64.60746993,
        44.36661812,  61.89318034,  68.71053472,  58.06475045,
        54.23002432,  45.64521772,  98.78951948,  77.56987908,
        92.51139605,  49.09106256,  61.09423884,  54.58602089,
        86.84726062,  31.67228227, 101.16426724,  62.7861187 ,
        99.28949733,  62.51725308,  39.89187057,  33.50981299,
        97.31175773,  82.66678232,  94.85657278,  29.99524988,
        66.91964487,  85.75182483,  81.1523426 ,  77.21561971,
        79.08992655,  46.31497701,  86.37921249,  65.61096138,
        45.03685301,  84.0355488 ,  59.74579292,  54.24075369,
        74.51184663,  49.51345283,  30.66615248,  63.84620129,
       102.20938001,  97.60049423,  59.36870491,  77.83589406,
        94.17014644,  91.63782641,  73.05982295,  69.22483354,
        39.10143825,  55.02918902,  59.24129892,  80.87492113,
        78.5594399 ,  58.87095202,  31.52650761,  37.43951013,
        80.62031628,  87.8908322 ,  86.9736507 ,  64.82254097,
        48.9260264 ,  73.87273822,  32.31315966,  88.05717757,
       100.92033267,  51.89844524,  66.14221749,  79.54210911,
        81.57681626,  97.17867287,  77.03863108,  81.84500658,
        82.89316102,  42.00733402,  94.20816318,  64.36112535,
        53.33239162,  86.48285954,  78.57203048,  48.47596726,
        36.3938819 ,  31.97145149,  50.50473407,  92.48583318,
        80.8438832 ,  39.32903493,  33.24380443,  63.11325419,
        50.27906649,  46.8136253 ,  43.16085657,  93.70774769,
        88.850665  ,  52.58694261,  96.77335471,  96.772105  ,
        42.01683792,  65.7522488 ,  46.85964737,  95.63887362,
        78.25233534,  70.47325484,  32.01373834,  64.77846828,
        78.0872578 ,  47.40746519,  88.58617776,  61.16712797,
        71.46562349,  68.59030328,  54.43433849,  76.7159058 ,
        97.34240033,  51.90413504,  96.51477687,  86.14514825,
        55.35285034,  31.57044967,  67.0065622 ,  88.53506648,
       100.50143168,  71.13486297,  86.09933255,  43.8723505 ,
        71.25619512,  97.31873215,  35.10992852,  75.67126905,
        40.56941284,  54.35369899,  75.92666029,  92.70871099,
        51.00561096,  32.86011274,  66.93073525,  57.5053555 ,
        43.72933746,  54.44610117, 101.48426474,  77.6269863 ,
        79.55381702, 102.45303272,  45.49848394,  73.03013117,
        87.45464678,  77.88614777, 102.26252326,  38.08595426])
```

In [33]:
```python
#Evaluate Linear Regression Model
print("Linear Regression Performance")
print("R² Score:", r2_score(y_test, y_pred_lr))
print("MAE:", mean_absolute_error(y_test, y_pred_lr))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))
```

```
Linear Regression Performance
R² Score: 0.9261780111559854
MAE: 4.777256806007201
RMSE: 5.592443454667665
```

In [34]:
```python
#Predicted Plot using scatter plot
plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred_lr)
plt.xlabel("Actual Stress Level")
plt.ylabel("Predicted Stress Level")
plt.title("Linear Regression: Actual vs Predicted")
plt.show()
```



In [35]:
```python
#Coefficients
coeff_df = pd.DataFrame({
    "Feature": X.columns,
    "Coefficient": lr.coef_
}).sort_values(by="Coefficient", ascending=False)

coeff_df
```

Out[35]:

	Feature	Coefficient
6	Cognitive_Load	19.129627
3	Bugs_Fixed	0.353855
4	AI_Usage_Hours	0.305281
9	Task_Duration_Hours	0.178956
1	Lines_of_Code	0.044882
0	Hours_Coding	-0.019413
8	Coffee_Intake	-0.087403
5	Sleep_Hours	-0.119673
11	Errors	-0.150598
10	Commits	-0.235803
2	Bugs_Found	-0.374346
7	Task_Success_Rate	-2.515326

In [10]:
```
 # Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

In [12]:
```
# scaling for the model
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [13]:
```
#Train Random Forest Regressor
rf = RandomForestRegressor(
    n_estimators=300,
    max_depth=10,
    random_state=42
)

rf.fit(X_train_scaled, y_train)
```

Out[13]:

▾ RandomForestRegressor

```
RandomForestRegressor(max_depth=10, n_estimators=300, random_state=42)
```

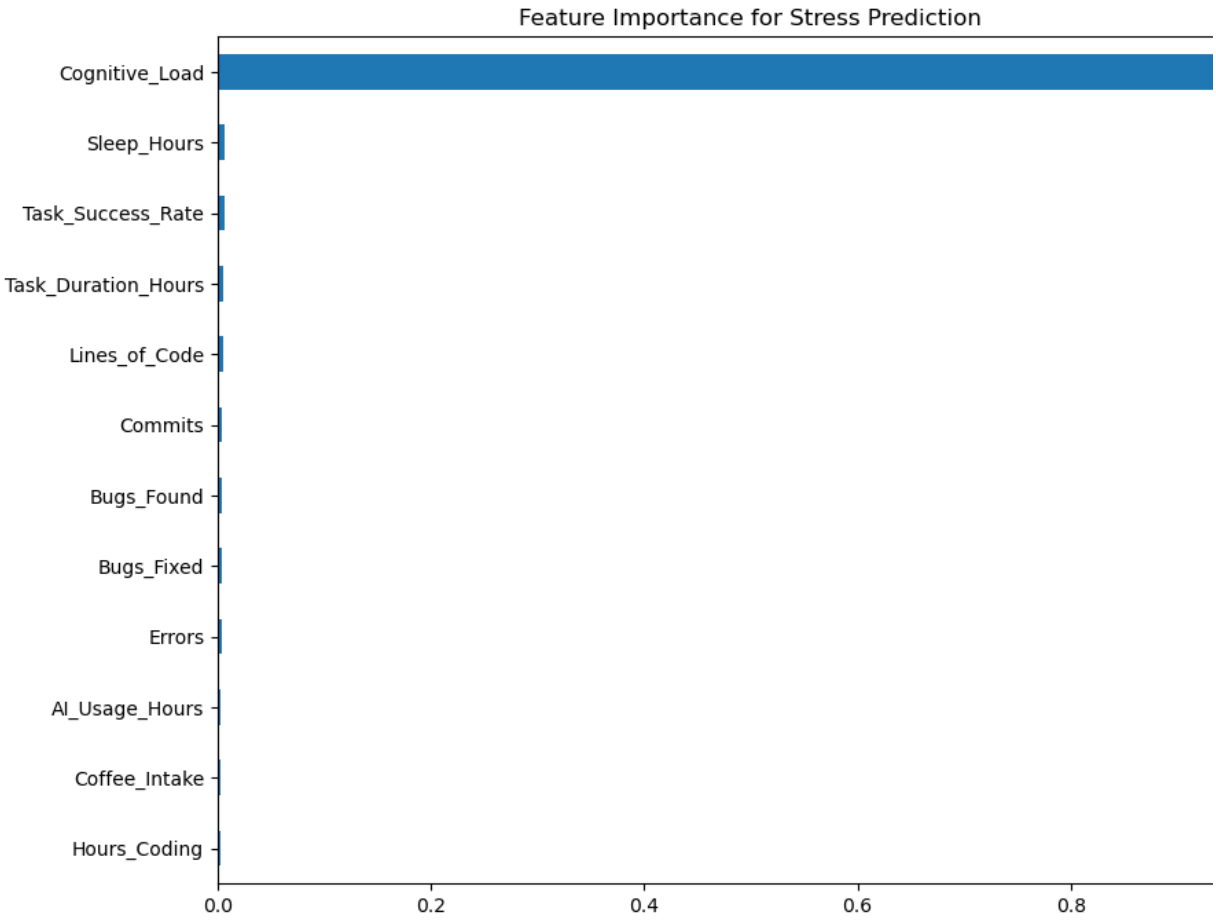
In [15]:
```
y_pred = rf.predict(X_test_scaled)      #Predictions of data
y_pred
```

Out[15]:
```
array([71.49253277, 92.87607472, 92.90910753, 71.07683946, 53.19653274,
       96.8136772 , 58.1788277 , 53.22639813, 60.05229478, 91.06132496,
       62.40516789, 69.98702906, 55.48754659, 41.88822995, 70.95437541,
       97.28727185, 76.42868086, 48.85436011, 35.03100872, 31.53360625,
       57.89662257, 85.20714453, 71.69421916, 72.16270529, 45.11315801,
       68.92376874, 35.03987766, 89.37698294, 54.54224955, 88.67384313,
       86.68252759, 50.94740177, 68.43150527, 32.78414463, 34.183495  ,
       60.87840245, 41.16104924, 39.65894646, 75.59047821, 76.22137243,
       97.65573502, 44.68404336, 33.62411174, 77.4877049 , 38.72289324,
       55.11961233, 98.74403325, 72.1861392 , 76.03437243, 55.99766412,
       79.22599309, 64.10215401, 42.69292818, 60.4879341 , 70.7824341 ,
       56.16660178, 56.07199097, 48.82473261, 96.64599876, 78.77590215,
       96.58789037, 51.30195479, 60.270818  , 56.13993184, 85.19223756,
       31.17386686, 98.6496715 , 63.74311411, 98.08664882, 58.54386904,
       39.49796563, 31.59615742, 97.27567829, 83.77943635, 95.26298329,
       31.63996629, 72.14898864, 89.68898512, 82.76890365, 79.07321146,
       79.48389464, 45.26544007, 88.39481858, 71.01991597, 44.07568779,
       88.2906065 , 56.21598514, 53.38696557, 72.04539933, 49.81322824,
       31.20696668, 60.25534369, 99.18569057, 98.2786478 , 58.52241032,
       76.59571361, 91.77101813, 93.84523362, 77.00351199, 70.18889957,
       37.38503612, 55.94854984, 59.00562322, 80.30589526, 83.18534462,
       56.84482661, 30.65439216, 37.65118602, 80.45630382, 89.51836953,
       89.17135225, 62.68230926, 50.24029819, 75.66673617, 31.30583576,
       91.2643214 , 97.58981971, 50.44335653, 69.60150571, 80.22781225,
       81.28853336, 97.1932572 , 78.60010806, 81.08508054, 84.10336493,
       39.09503127, 92.0109279 , 62.53921129, 54.86384232, 89.25956363,
       80.57761567, 44.86523812, 36.44696706, 35.61197455, 49.7188706 ,
       97.19857921, 83.91645169, 35.32764049, 31.5276524 , 63.10381491,
       49.49597598, 47.74564254, 38.96823362, 90.81400577, 86.00539738,
       52.07636785, 99.01794304, 95.18636802, 38.71432641, 71.21728254,
       46.11593896, 92.19249086, 77.5006107 , 70.64799448, 33.28624551,
       63.57983268, 79.41314387, 46.62736342, 87.8151474 , 60.52792711,
       70.66774645, 69.51927687, 54.85194442, 75.03288686, 97.35539981,
       51.74043494, 98.34682669, 87.10752784, 56.2217018 , 31.31615239,
       68.99776976, 88.63346919, 97.80863209, 73.23036694, 86.90934569,
       43.98173371, 70.9232382 , 96.55241226, 32.37529537, 76.21139408,
       40.50259542, 52.82806444, 77.7816078 , 92.72907783, 51.92077411,
       31.10812825, 69.31634603, 57.7122418 , 40.67687829, 57.19348187,
       98.09566655, 79.76373873, 78.87795722, 98.6342319 , 46.30515645,
       70.5594308 , 89.64359987, 78.23597156, 96.89285462, 39.05299731])
```

In [18]:
```
#Model Evaluation
print("R² Score:", r2_score(y_test, y_pred))
print("MAE:", mean_absolute_error(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
R² Score: 0.9228393680697207
MAE: 4.78735700163963
RMSE: 5.717505865636008
```

In [17]:
```
#Feature Importance using barPlot
feat_importances = pd.Series(rf.feature_importances_, index=X.columns)
feat_importances.sort_values().plot(kind='barh', figsize=(10,8))
plt.title("Feature Importance for Stress Prediction")
plt.show()
```



In [19]:
```python
#Predict Stress Level
new_data = pd.DataFrame({
    "Hours_Coding": [8],
    "Lines_of_Code": [400],
    "Bugs_Found": [10],
    "Bugs_Fixed": [6],
    "AI_Usage_Hours": [4],
    "Sleep_Hours": [5.5],
    "Cognitive_Load": [70],
    "Task_Success_Rate": [50],
    "Coffee_Intake": [3],
    "Task_Duration_Hours": [9],
    "Commits": [20],
    "Errors": [3]
})

new_scaled = scaler.transform(new_data)
prediction = rf.predict(new_scaled)

print("Predicted Stress Level:", prediction[0])
```

```
Predicted Stress Level: 80.62089634463517
```

In [ ]: