CA1

Super Life Insurance

Super Life has got you covered!

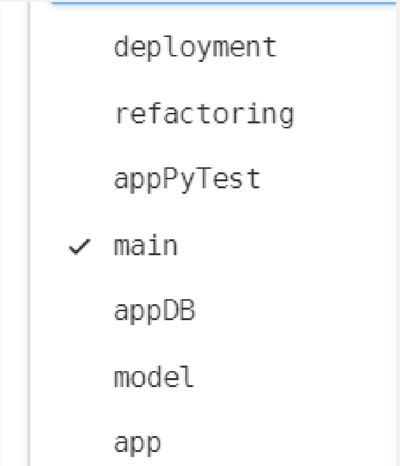
What is super life: Super Life is an AI application, that will indicate to you what is the most reasonable price for a year's worth of health insurance!

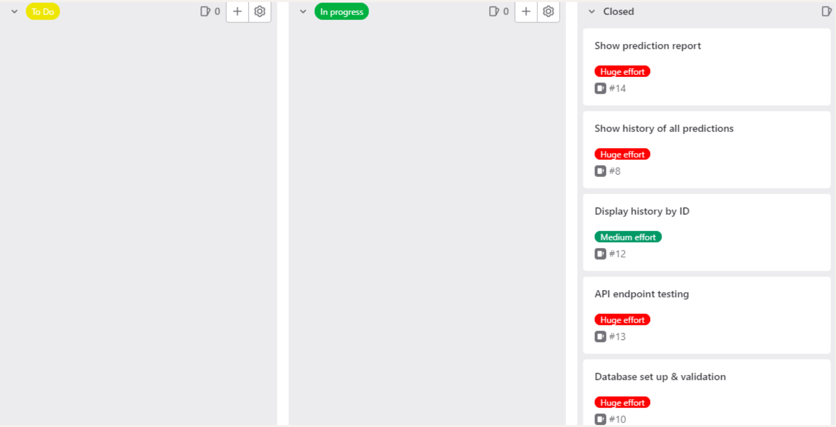
Aim: It enables people to make a more informed decision prior to purchasing insurance premium. Such can help them to make better financial decisions

Lim Hur 2112589
DAAA2A02

SCRUM board

- All issues are closed as we have completed.
- We have 6 branches, but did not do deployment.
- Main branch,
- Model branch for modelling,
- App branch for UI and backend without database
- appDB branch for app with databases implemented.
- Testing for pytests
- Refactoring to improve some codes
- Deployment (didn't do)
- Git repo link: https://gitlab.com/2589-st1505/ca1-daaa2b02-2112589-limhur/-/tree/main



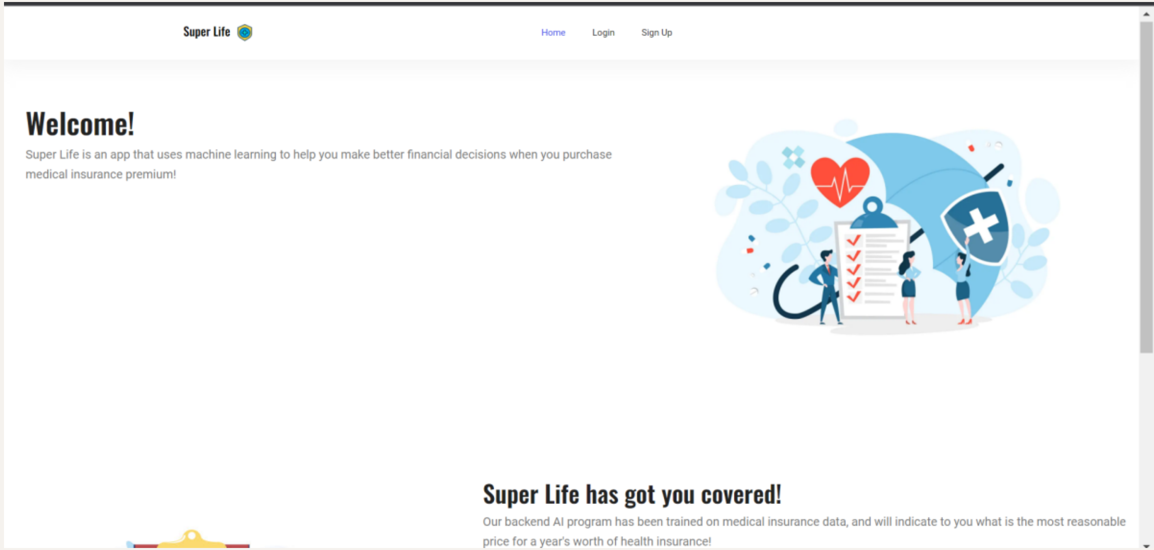


Website

- We have 5 pages:

Home, Login, Prediction, History

- Our homepage is shown below, with a simple welcome message.
- To make the best out of this application, only intrapolation is allowed, that means only values within the range used to train the model are valid.



Model implementation

- Dataset from kaggle, which is about medical insurance premium.
- Contains user’s health information like whether they have any diseases, etc.
- Columns are on the right, most are binary (1 and 0) and some are continuous like weight/height.
- Given this, our task is to develop regression model for predicting insurance prices.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 986 entries, 0 to 985
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      986 non-null    int64
 1   Diabetes                 986 non-null    int64
 2   BloodPressureProblems    986 non-null    int64
 3   AnyTransplants           986 non-null    int64
 4   AnyChronicDiseases       986 non-null    int64
 5   Height                   986 non-null    int64
 6   Weight                   986 non-null    int64
 7   KnownAllergies           986 non-null    int64
 8   HistoryOfCancerInFamily  986 non-null    int64
 9   NumberOfMajorSurgeries   986 non-null    int64
 10  PremiumPrice             986 non-null    int64
dtypes: int64(11)
```

Feature engineering

- Feature engineering: BMI,
- Feature scaling using standard scaler

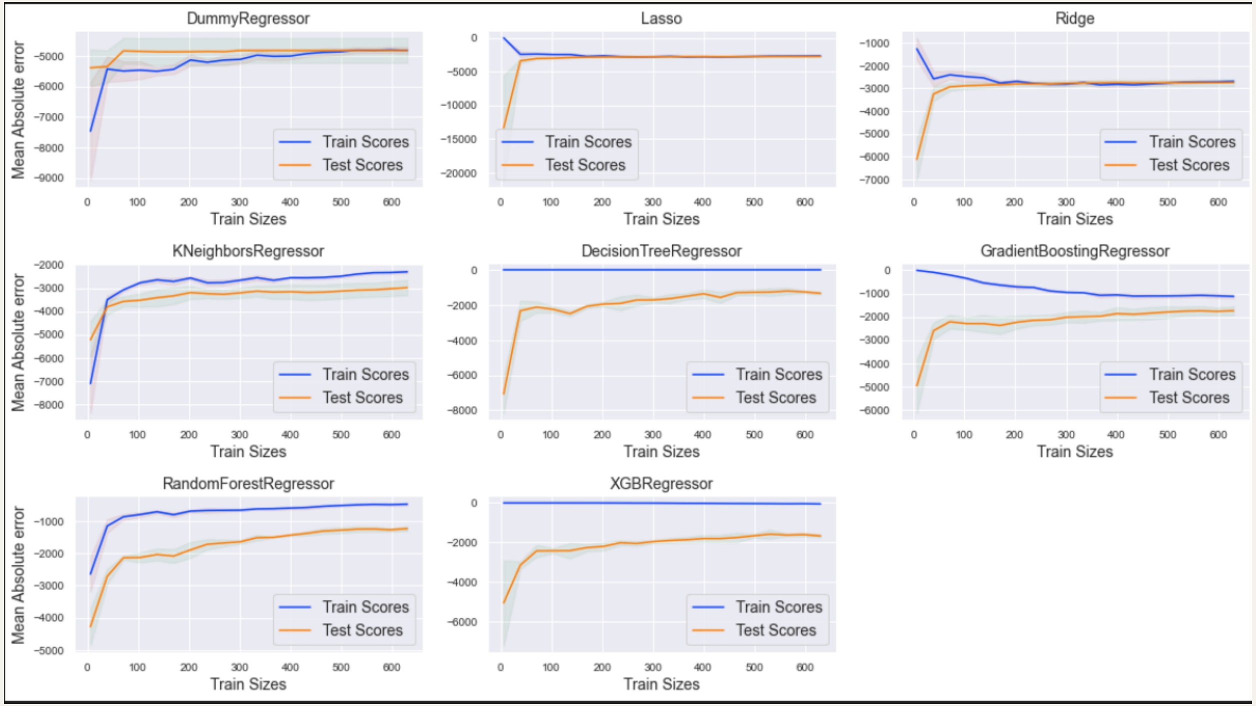
```
med_df['BMI'] = med_df['Weight'] / ((med_df['Height']/100) ** 2 )
```

Feature engineering

- BMI - Weight / (height **2) ; (height in meters)
  - Acts as an indicator to tell whether someone is obese, acceptable weight or underweight. This can be an indicator on how healthy the person weight is.
  - Could provide more useful information into the model, to produce more accurate results
  - Insurance organizations do use this information to gauge how healthy one's weight is.

```
custom_scaling = ColumnTransformer(
    transformers=[('scaler', StandardScaler(), list(X_train.columns) )],
    remainder='passthrough'
)
preprocess = Pipeline([
    ('Scaling', custom_scaling)
])
```

Model implementation



Model implementation

1. Linear Models
- Lasso and Ridge regression generally do not overfit, and have pretty low variance given the small difference in CV and train MAE scores.
- They do however underfit as they may not be complex enough and generally performs poorly
- These models pale in comparison to ensemble models in terms of MAE- One possible improvement is to introduce PolynomialFeatures() which generate polynomial and interaction features

2. Distance based models
- KNeighborsRegressor might be suffering from slight underfitting, and has a decent average MAE score.Slightly overfitting given the larger difference in errors in CV and training sets
- One improvement approach to take could be changing the scalers

3. Tree Models
- Severely overfits given the large gap between average train MAE and CV MAE. We observe a difference of close to a thousand MAE.
- Performs better than linear models and distance models in terms of cross validation mean absolute error
- One improvement could be further tuning model by optimizing hyperparameters

4. Ensembles
- Gradient Boosting and Random forests perform very well, suffering from slight overfitting.
- RandomForest posses low bias, indicated by low consistent errors from CV sets.
- Gradient boosting generally have pretty high bias, which seems like it is underfitting.
- One improvement approach to take is to tune hyper parameters, such as max_depth or learning_rate for gradient boostings.

5. XGBoost
- XGBoost model severely overfits, with large deviation between average cv mae and train MAE.

	DummyRegressor	Lasso	Ridge	KNeighborsRegressor	DecisionTreeRegressor	GradientBoostingRegressor	RandomForestRegressor	XGBRegressor
train_mae	-4810.940429	-2679.632829	-2683.394443	-2290.417729	0.000000	-1110.642329	-469.794986	-47.555971
cv_mae	-4798.440457	-2705.737200	-2705.248700	-2906.154500	-1137.707500	-1670.333800	-1258.631929	-1549.727886
train_mape	-0.215371	-0.116214	-0.116457	-0.102157	0.000000	-0.045171	-0.019757	-0.002043
cv_mape	-0.214971	-0.117300	-0.117500	-0.128586	-0.048300	-0.068557	-0.052057	-0.063771
cv_r2	-0.044500	0.618457	0.620586	0.518514	0.590357	0.755414	0.748171	0.716029

Model implementation

Random Forests were chosen, as it overfits the least, and posses low bias/variance, which indicates that it is able to generalize to new examples well

Perform hyperparameter optimization to improve model. Model has improved in terms of MAE.

After final evaluation out model achieves around 1000 MAE on test set, and which is quite good. Model does not overfit given its small difference between CV and train scores.

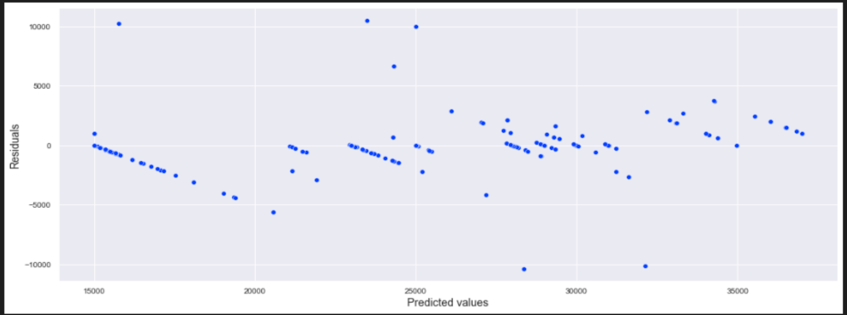
Age is the most important feature shown In the feature importance plot.

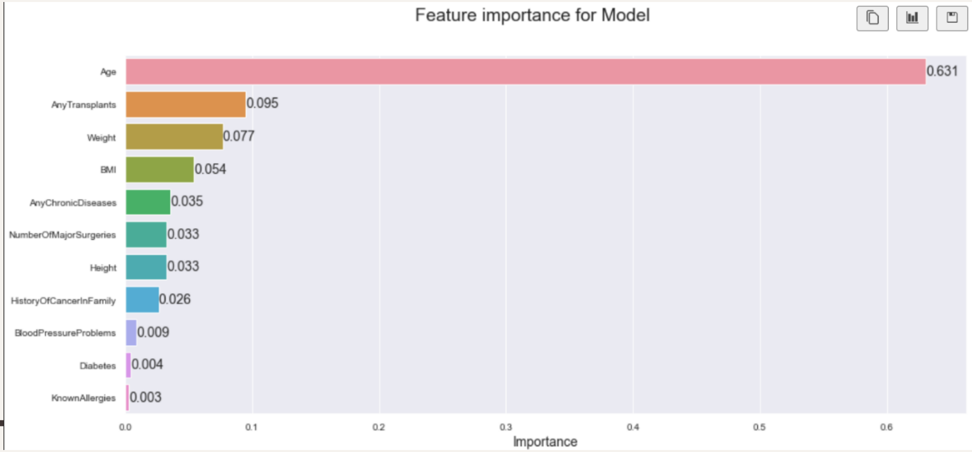
Tuning score:

```
1  final_model.best_score_
159]
-1091.395534805248
```

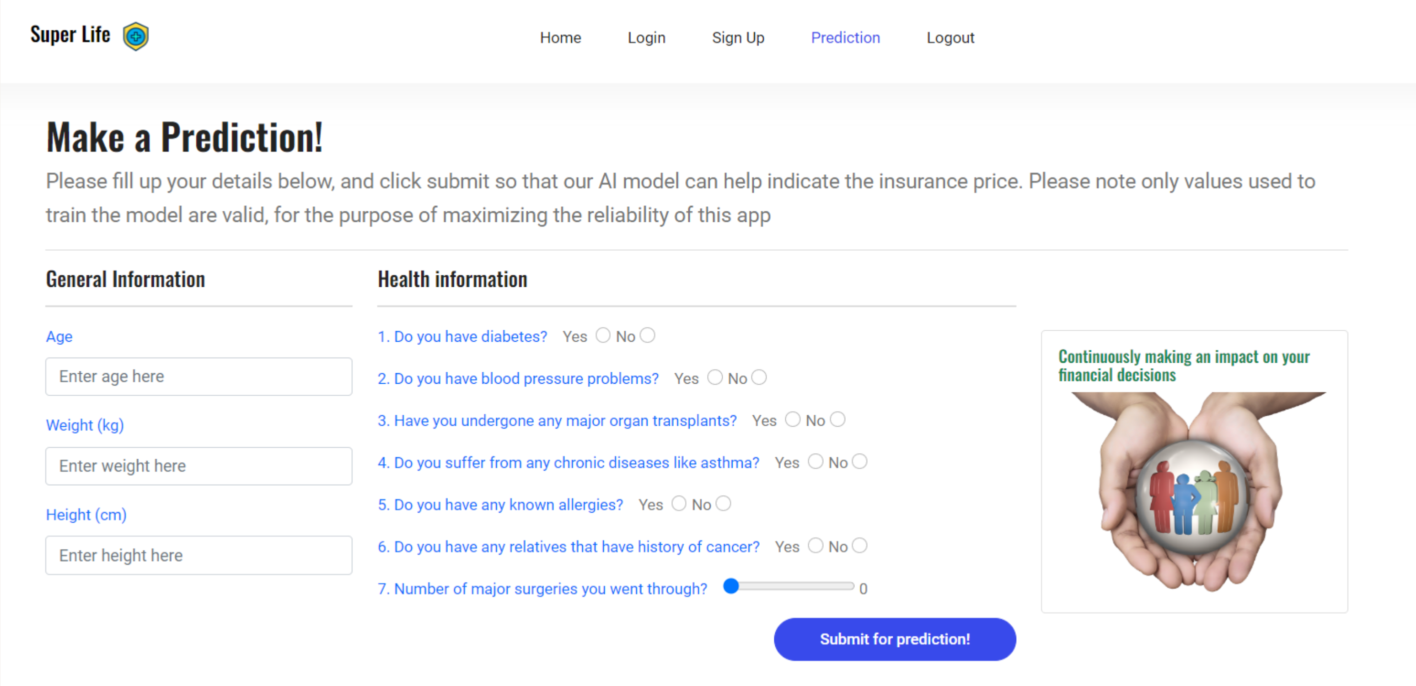
	MAE	MAPE
0	1098.514299	0.043916

```
Mean Absolute percentage error: 0.04391595407761472
```



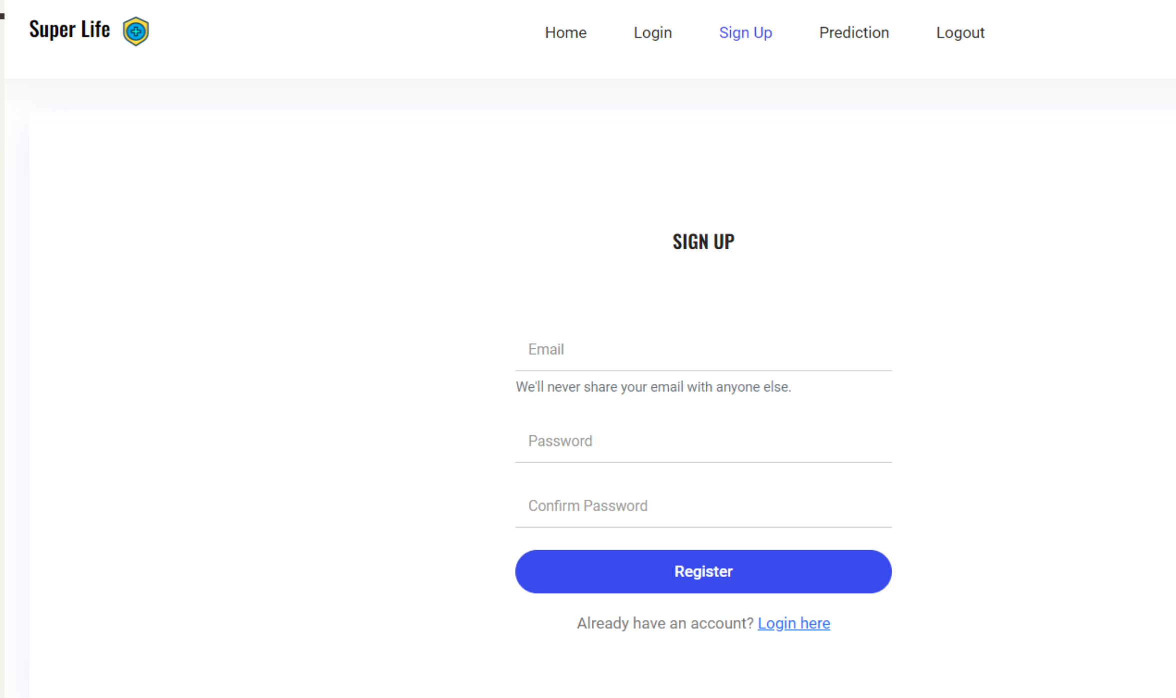


Prediction page



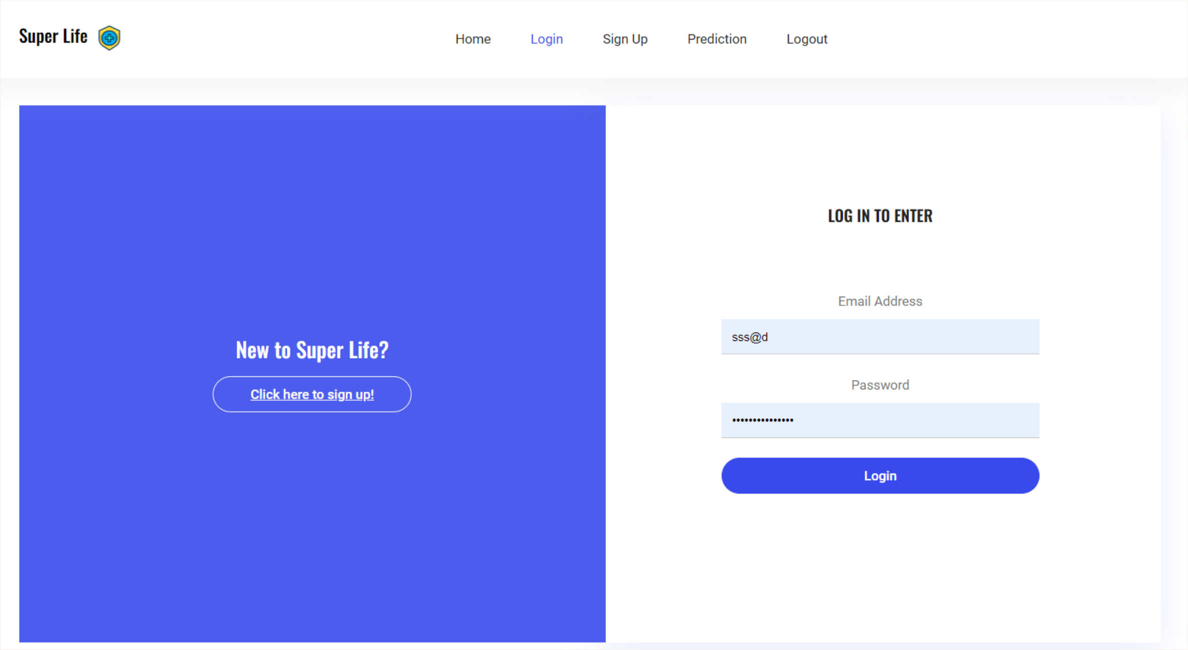
- Users can enter their health details and general information for the model to process the inputs.
- Only the range values used to train the model are valid in this application to maximize realibility

Sign up page



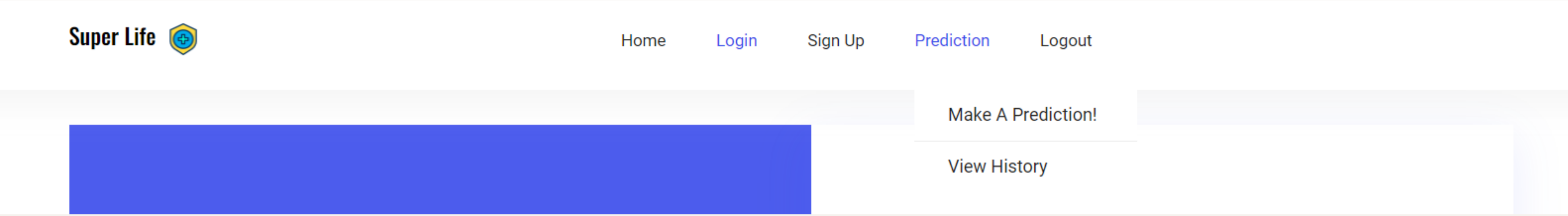
- Allows users to enter 3 fields: email, password and confirm password
- Upon successful creation, we store the details in the database ‘User’ table. This is so that we can show the correct history for the correct user.
- The website will prompts a success message after signing up successfully as well.

Login page

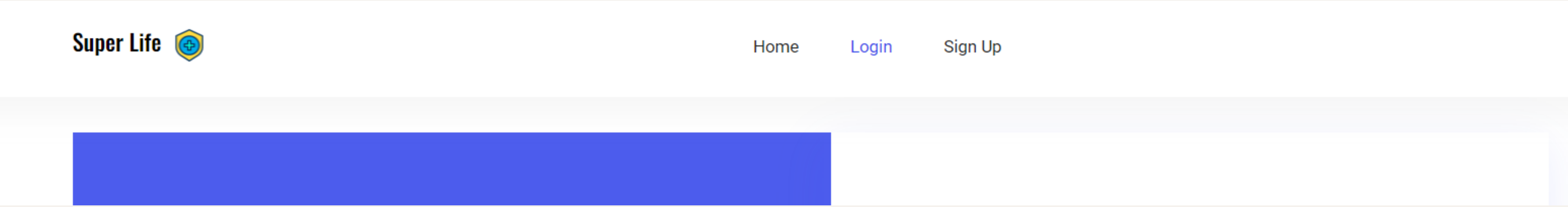


- Allows users to enter 3 fields: email, password
- Upon successful login, user gets redirected to home page, and they can view the prediction/history page. Use of flask_login to handle the login task.

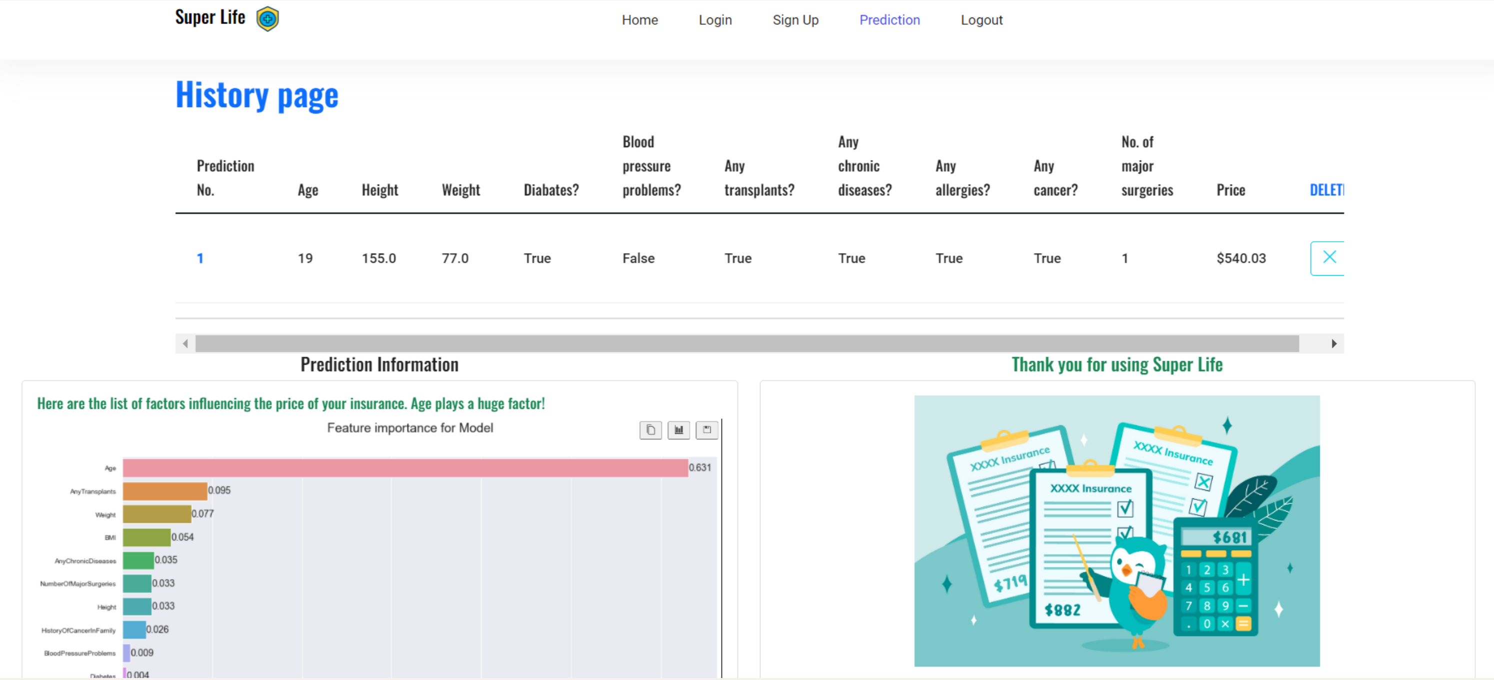
Flask login



- When a user is logged in they can see prediction and history page. However if user is not logged in, they can only see login and sign up page

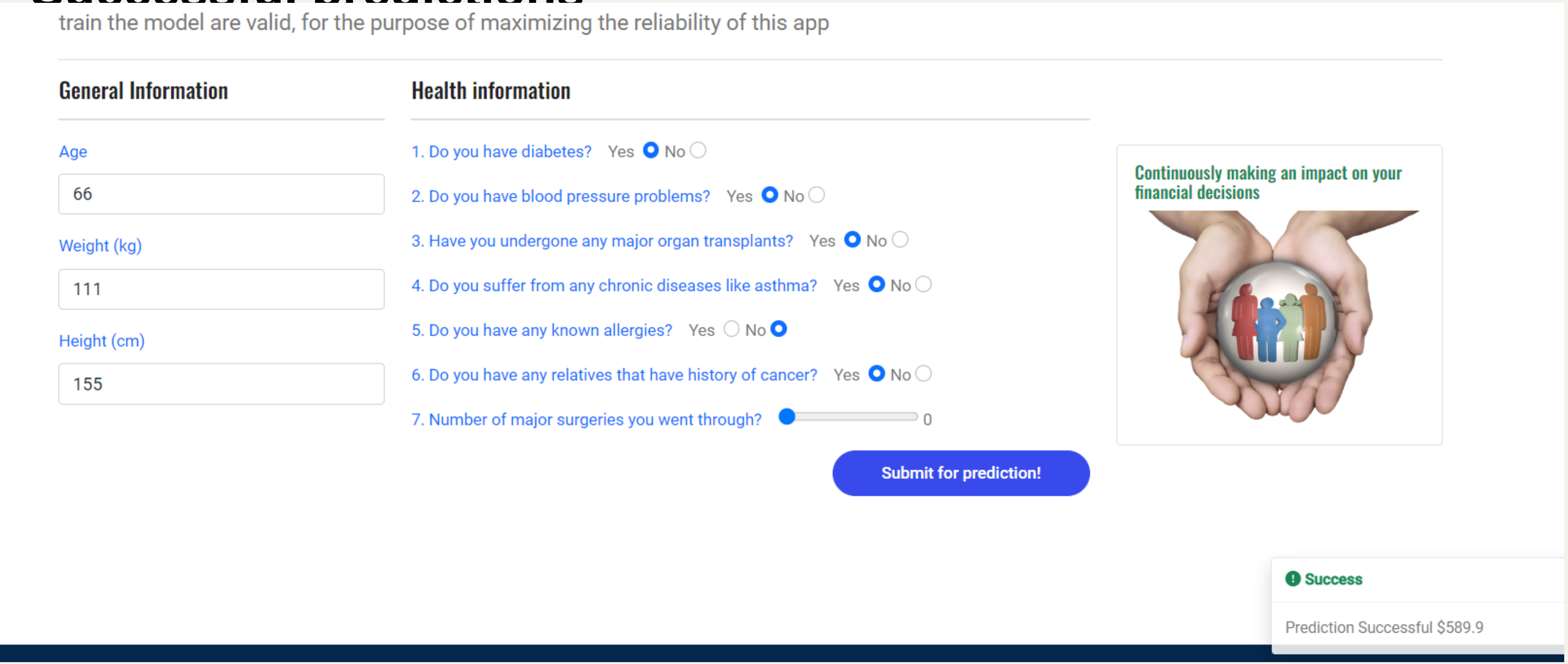


History page



- History page shows past predictions and shows feature importance for model. Allows user to delete their predictions through the delete button.

Successful predictions



- We have a success message at the bottom to prompt the successful prediction