Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df = pd.read_csv("FineTech_appData.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	user	first_open	dayofweek	hour	age	screen
0	235136	2012-12-27 02:14:51.273	3	02:00:00	23	idscreen,joinscreen,Cycle,product_review,Sca
1	333588	2012-12-02 01:16:00.905	6	01:00:00	24	joinscreen,product_review,product_review2,Sc
2	254414	2013-03-19 19:19:09.157	1	19:00:00	23	Splash,Cycle,
3	234192	2013-07-05 16:08:46.354	4	16:00:00	28	product_review,Home,product_review,Loan3,Fir
4	51549	2013-02-26 18:50:48.661	1	18:00:00	31	idscreen,joinscreen,Cycle,Credit3Container,S

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
user                        0
first_open                  0
dayofweek                   0
hour                        0
age                         0
screen_list                 0
numscreens                  0
minigame                    0
used_premium_feature        0
enrolled                    0
enrolled_date           18926
liked                       0
dtype: int64
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   user                  50000 non-null  int64 
 1   first_open            50000 non-null  object
 2   dayofweek             50000 non-null  int64 
 3   hour                  50000 non-null  object
 4   age                   50000 non-null  int64 
 5   screen_list           50000 non-null  object
 6   numscreens            50000 non-null  int64 
 7   minigame              50000 non-null  int64 
 8   used_premium_feature  50000 non-null  int64 
 9   enrolled              50000 non-null  int64 
 10  enrolled_date         31074 non-null  object
 11  liked                 50000 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 4.6+ MB
```

In [6]:

```
df.describe()
```

Out[6]:

	user	dayofweek	age	numscreens	minigame	used_premium
count	50000.000000	50000.000000	50000.00000	50000.000000	50000.000000	5000(
mean	186889.729900	3.029860	31.72436	21.095900	0.107820	(
std	107768.520361	2.031997	10.80331	15.728812	0.310156	(
min	13.000000	0.000000	16.00000	1.000000	0.000000	(
25%	93526.750000	1.000000	24.00000	10.000000	0.000000	(
50%	187193.500000	3.000000	29.00000	18.000000	0.000000	(
75%	279984.250000	5.000000	37.00000	28.000000	0.000000	(
max	373662.000000	6.000000	101.00000	325.000000	1.000000	

In [7]:

```
df2 = df.copy()
```

Droping string data for data visualization

In [8]:

```
df2.drop(["user","first_open","screen_list","enrolled_date"],axis=1,inplace=True)
```

In [9]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   dayofweek             50000 non-null  int64 
 1   hour                  50000 non-null  object
 2   age                   50000 non-null  int64 
 3   numscreens            50000 non-null  int64 
 4   minigame              50000 non-null  int64 
 5   used_premium_feature  50000 non-null  int64 
 6   enrolled              50000 non-null  int64 
 7   liked                 50000 non-null  int64 
dtypes: int64(7), object(1)
memory usage: 3.1+ MB
```

Converting datatype of hour to int

In [10]:

```
df2['hour'] = df2.hour.str.slice(1,3).astype(int)
```

In [11]:

```
df2.info()
```

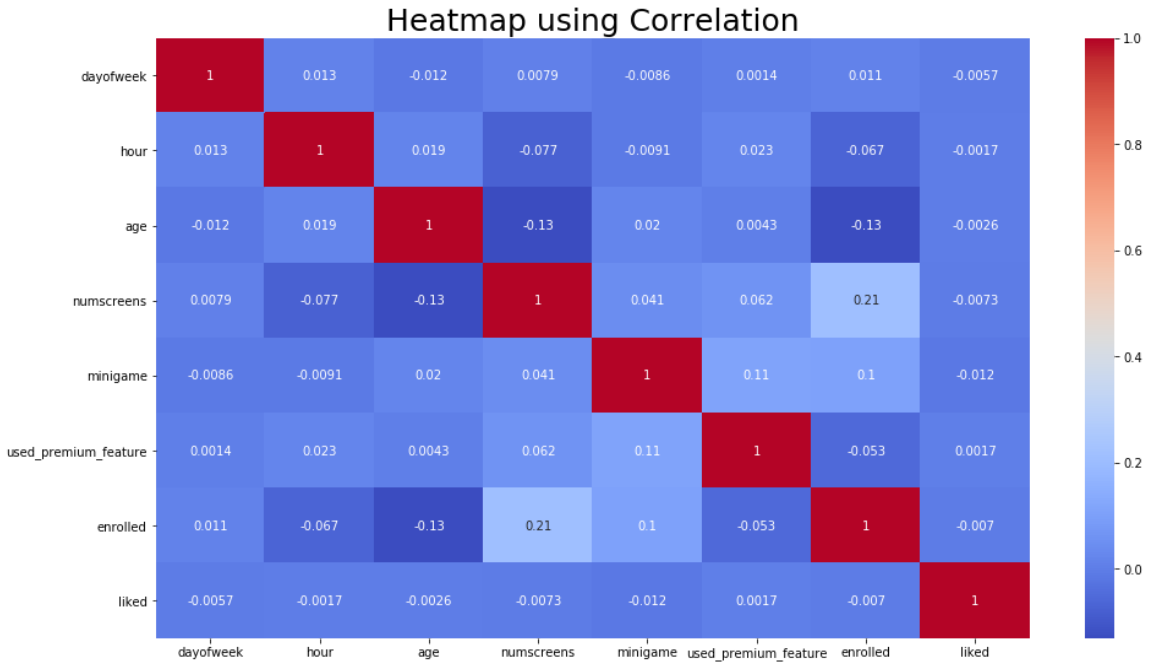
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   dayofweek             50000 non-null  int64 
 1   hour                  50000 non-null  int32 
 2   age                   50000 non-null  int64 
 3   numscreens            50000 non-null  int64 
 4   minigame              50000 non-null  int64 
 5   used_premium_feature  50000 non-null  int64 
 6   enrolled              50000 non-null  int64 
 7   liked                 50000 non-null  int64 
dtypes: int32(1), int64(7)
memory usage: 2.9 MB
```

In [12]:

```
plt.figure(figsize=(16,9))
sns.heatmap(df2.corr(),annot=True,cmap="coolwarm")
plt.title("Heatmap using Correlation",fontsize=25)
```

Out[12]:

```
Text(0.5, 1, 'Heatmap using Correlation')
```



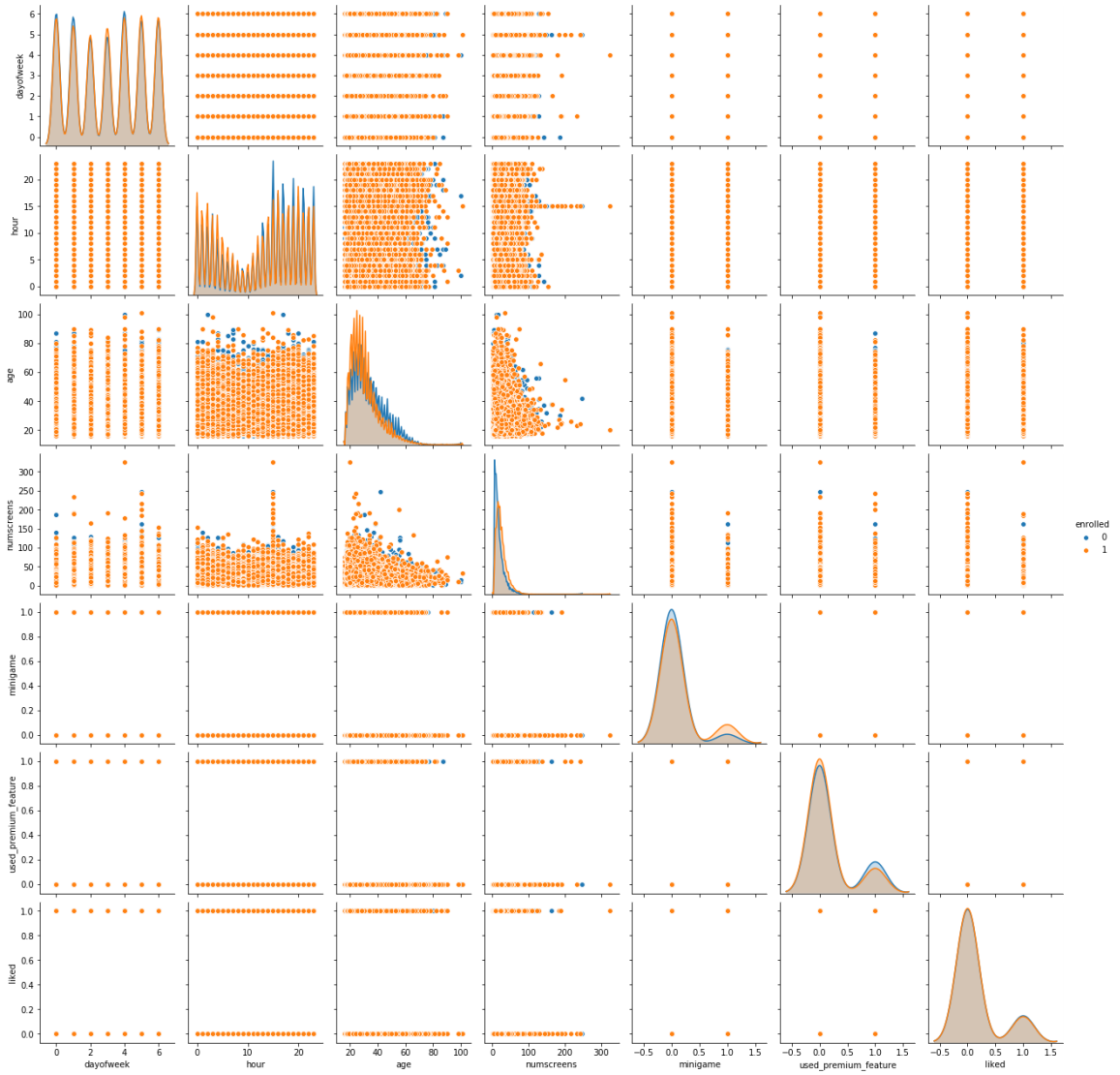
The heatmap above shows there is little correlation between ‘numscreens’ and ‘enrolled’. It means that those customers saw more screen they are taking premium app. There is a slight correlation between ‘minigame’ with ‘enrolled’ and ‘used_premium_feature’. The slightly negative correlation between ‘age’ with ‘enrolled’ and ‘numscreens’. It means that older customers do not use the premium app and they don’t see multiple screens.

In [13]:

```
sns.pairplot(df2, hue='enrolled',diag_kws={'bw': 0.2})
```

Out[13]:

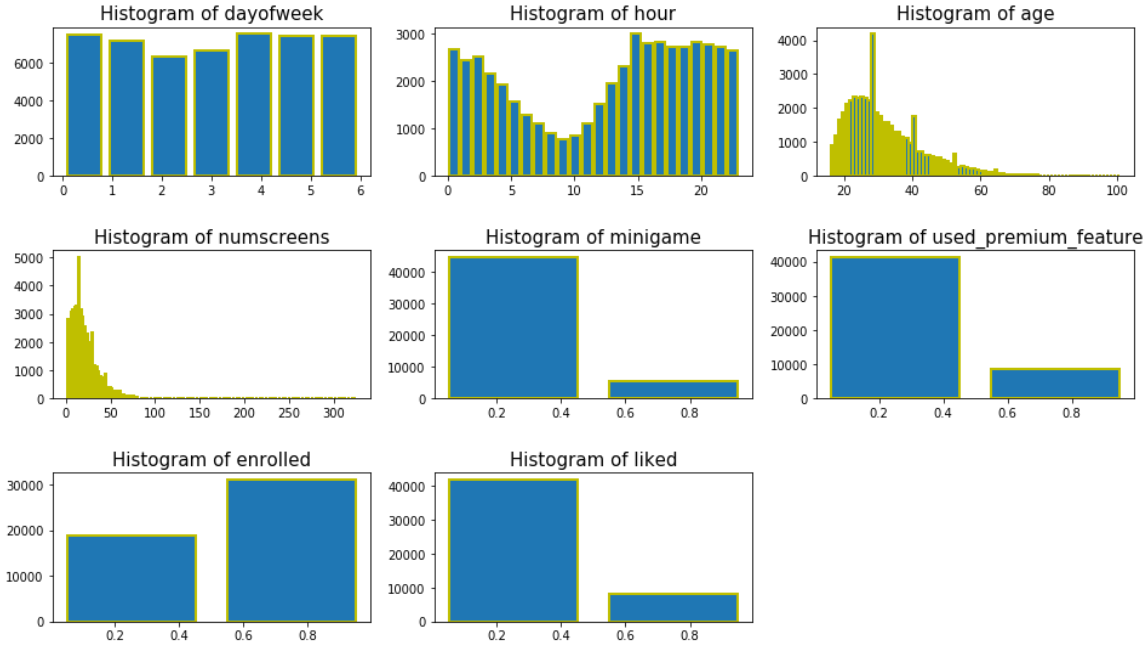
```
<seaborn.axisgrid.PairGrid at 0x1bcda5d0a08>
```



In pair plot we can see, the maximum features have two values like 0 and 1 and orange dots show the enrolled customer's features

In [14]:

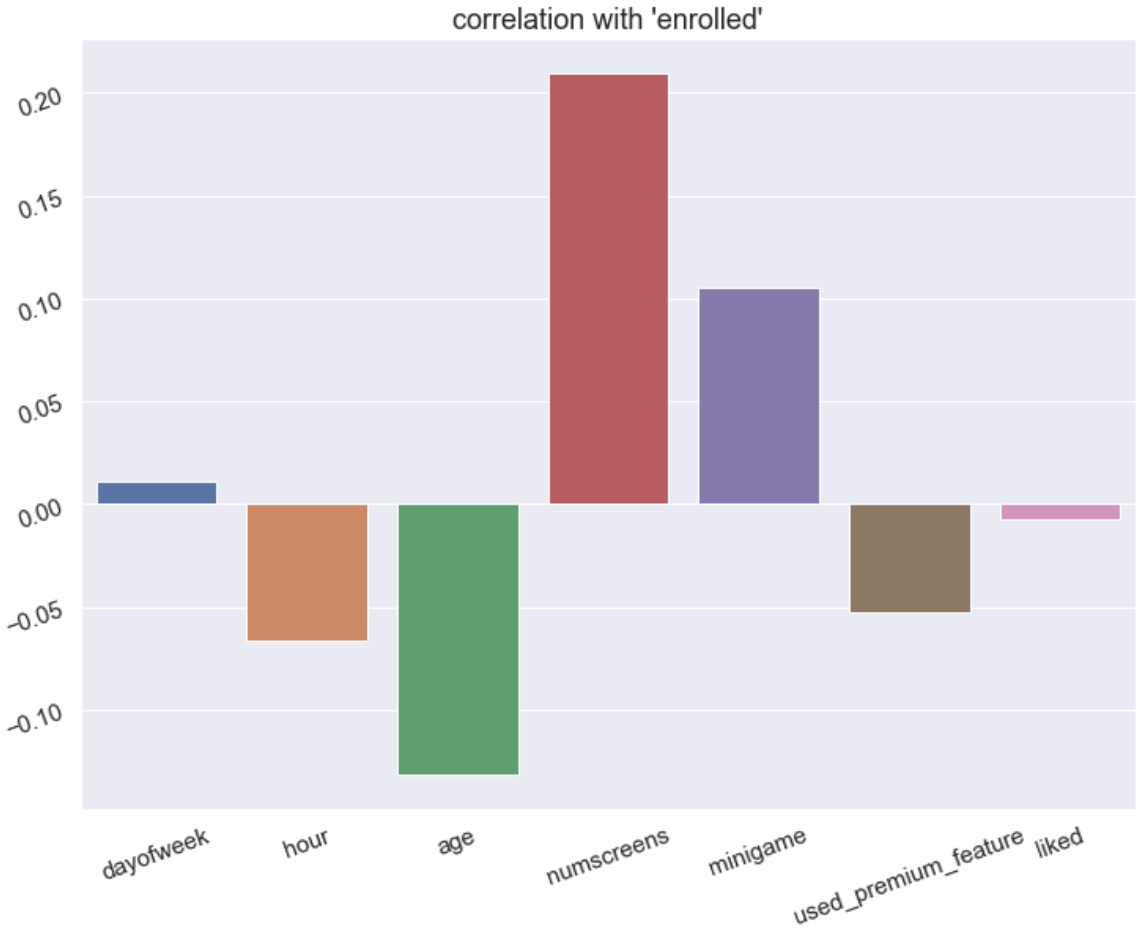
```
plt.figure(figsize=(16,9))
features=df2.columns
for i,j in enumerate(features):
    plt.subplot(3,3,i+1)
    plt.title("Histogram of {}".format(j),fontsize=15)
    bins=len(df2[j].unique())
    plt.hist(df2[j],bins=bins,rwidth=0.8,edgecolor="y",linewidth=2)
plt.subplots_adjust(hspace=0.5)
```



In the above histogram, we can see minigame, used_primium_feature, enrolled, and like they have only two values and how they distributed.The histogram of 'dayofweek' shows, on Tuesday and Wednesday slightly fewer customer registered the app.The histogram of 'hour' shows the less customer register on the app around 10 AM.The 'age' histogram shows, the maximum customers are younger.The 'numsreens' histogram shows the few customers saw more than 40 screens.

In [15]:

```
sns.set()
plt.figure(figsize=(12,9))
plt.title("correlation with 'enrolled'",fontsize=18)
df3 = df2.drop(["enrolled"],axis=1)
ax= sns.barplot(df3.columns,df3.corrwith(df2.enrolled))
ax.tick_params(labelsize=15,labelrotation=20,color="k")
```



We saw the heatmap correlation matrix but this was not showing correlation clearly but you can easily understand which feature is how much correlated with 'enrolled' feature using the above barplot.The 'numscreens' and 'minigame' is strongly positively correlated with 'enrolled' feature than other feature.The 'hour', 'age' and 'used_premium_feature' are strongly negatively correlated with the 'enrolled' feature.

Now, convert date data into timestamp

In [19]:

```
df4 = df2.copy()
```

In [22]:

```
df4.head()
```

Out[22]:

	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	liked
0	3	2	23	15	0	0	0	0
1	6	1	24	13	0	0	0	0
2	1	19	23	3	0	1	0	1
3	4	16	28	40	0	0	1	0
4	1	18	31	32	0	0	1	1

In [16]:

```
from dateutil import parser
```

In [17]:

```
df["first_open"] = [parser.parse(i) for i in df["first_open"]]
```

In [23]:

```
df["enrolled_date"] = [parser.parse(i) if isinstance(i,str) else i for i in df["enrolle
d_date"]]
```

In [25]:

```
df['hour'] = df.hour.str.slice(1,3).astype(int)
```

In [26]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   user                  50000 non-null  int64         
 1   first_open            50000 non-null  datetime64[ns]
 2   dayofweek             50000 non-null  int64         
 3   hour                  50000 non-null  int32         
 4   age                   50000 non-null  int64         
 5   screen_list           50000 non-null  object        
 6   numscreens            50000 non-null  int64         
 7   minigame              50000 non-null  int64         
 8   used_premium_feature  50000 non-null  int64         
 9   enrolled              50000 non-null  int64         
 10  enrolled_date         50000 non-null  datetime64[ns]
 11  liked                 50000 non-null  int64         
dtypes: datetime64[ns](2), int32(1), int64(8), object(1)
memory usage: 4.4+ MB
```

In [42]:

```
df5 = df.copy()
```

In [43]:

```
df5.head()
```

Out[43]:

	user	first_open	dayofweek	hour	age	screen_lis
0	235136	2012-12-27 02:14:51.273	3	2	23	idscreen,joinscreen,Cycle,product_review,ScanP.
1	333588	2012-12-02 01:16:00.905	6	1	24	joinscreen,product_review,product_review2,Scan.
2	254414	2013-03-19 19:19:09.157	1	19	23	Splash,Cycle,Loa
3	234192	2013-07-05 16:08:46.354	4	16	28	product_review,Home,product_review,Loan3,Finan.
4	51549	2013-02-26 18:50:48.661	1	18	31	idscreen,joinscreen,Cycle,Credit3Container,Sca.

We'll subtract enrolled date from first_open to check how much time enrolled in the premium feature app after registration

In [44]:

```
df5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   user                  50000 non-null  int64         
 1   first_open            50000 non-null  datetime64[ns]
 2   dayofweek             50000 non-null  int64         
 3   hour                  50000 non-null  int32         
 4   age                   50000 non-null  int64         
 5   screen_list           50000 non-null  object        
 6   numscreens            50000 non-null  int64         
 7   minigame              50000 non-null  int64         
 8   used_premium_feature  50000 non-null  int64         
 9   enrolled              50000 non-null  int64         
 10  enrolled_date         50000 non-null  datetime64[ns]
 11  liked                 50000 non-null  int64         
 12  time to enroll        50000 non-null  float64       
dtypes: datetime64[ns](2), float64(1), int32(1), int64(8), object(1)
memory usage: 4.8+ MB
```

In [46]:

```
df5["time to enroll"] = (df5['enrolled_date']-df5["first_open"]).astype('timedelta64
[h]')
```

In [51]:

```
df5["time to enroll"].unique()
```

Out[51]:

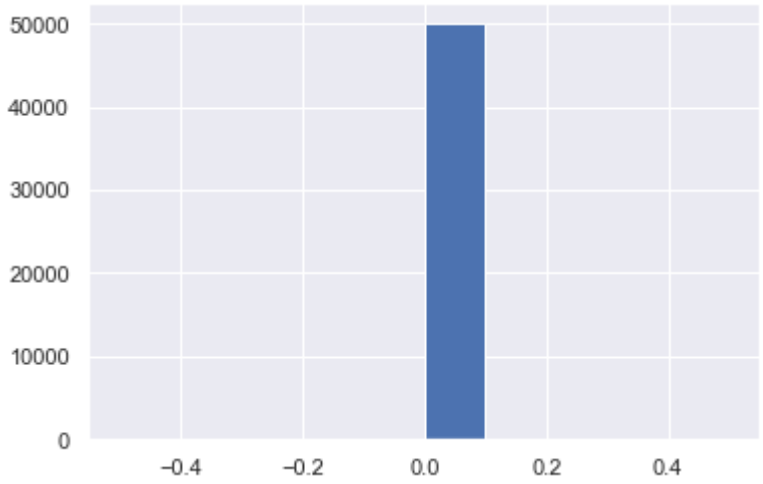
```
array([0.])
```

In [48]:

```
plt.hist(df5['time to enroll'].dropna())
```

Out[48]:

```
(array([    0.,     0.,     0.,     0.,     0., 50000.,     0.,     0.,
            0.,     0.]),
 array([-0.5, -0.4, -0.3, -0.2, -0.1,  0. ,  0.1,  0.2,  0.3,  0.4,  0.
5]),
 <a list of 10 Patch objects>)
```

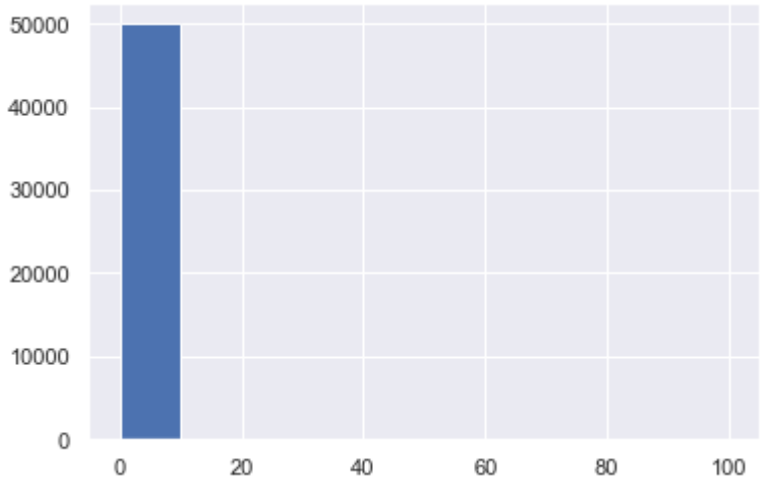


In [57]:

```
plt.hist(df5['time to enroll'].dropna(), range = (0,100))
```

Out[57]:

```
(array([50000.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
            0.,     0.]),
 array([  0.,  10.,  20.,  30.,  40.,  50.,  60.,  70.,  80.,  90., 10
0.]),
 <a list of 10 Patch objects>)
```



In [58]:

```
df5.loc[df['time to enroll'] > 48, 'enrolled'] = 0
```

we'll drop 'time to enroll', 'enrolled_date', 'first_open' feauture as they are not correlated to determine result

In [60]:

```
df5.drop(columns = ['time to enroll', 'enrolled_date', 'first_open'], inplace=True)
```

As next step we are going to import screen list,make them into columns and assign 0 and 1 according to screen usuage

In [61]:

```
df_screen_list =pd.read_csv("top_screens.csv").top_screens.values
```

In [62]:

```
df_screen_list
```

Out[62]:

```
array(['Loan2', 'location', 'Institutions', 'Credit3Container',
       'VerifyPhone', 'BankVerification', 'VerifyDateOfBirth',
       'ProfilePage', 'VerifyCountry', 'Cycle', 'idscreen',
       'Credit3Dashboard', 'Loan3', 'CC1Category', 'Splash', 'Loan',
       'CC1', 'RewardsContainer', 'Credit3', 'Credit1', 'EditProfile',
       'Credit2', 'Finances', 'CC3', 'Saving9', 'Saving1', 'Alerts',
       'Saving8', 'Saving10', 'Leaderboard', 'Saving4', 'VerifyMobile',
       'VerifyHousing', 'RewardDetail', 'VerifyHousingAmount',
       'ProfileMaritalStatus', 'ProfileChildren ', 'ProfileEducation',
       'Saving7', 'ProfileEducationMajor', 'Rewards', 'AccountView',
       'VerifyAnnualIncome', 'VerifyIncomeType', 'Saving2', 'Saving6',
       'Saving2Amount', 'Saving5', 'ProfileJobTitle', 'Login',
       'ProfileEmploymentLength', 'WebView', 'SecurityModal', 'Loan4',
       'ResendToken', 'TransactionList', 'NetworkFailure', 'ListPicker'],
      dtype=object)
```

In [69]:

```
df5['screen_name'] =  df['screen_list'].astype(str)+','
```

In [70]:

```
df5.screen_name.unique()
```

Out[70]:

```
array(['idscreen,joinscreen,Cycle,product_review,ScanPreview,VerifyDateOfB
irth,VerifyPhone,VerifyToken,ProfileVerifySSN,Loan2,Settings,ForgotPasswor
d,Login,',
       'joinscreen,product_review,product_review2,ScanPreview,VerifyDateOf
Birth,location,VerifyCountry,VerifyPhone,VerifyToken,Institutions,Loan2,',
       'Splash,Cycle,Loan,', ...,
       'joinscreen,product_review,product_review2,ScanPreview,VerifyCountr
y,VerifyPhone,VerifyToken,VerifyDateOfBirth,location,Home,',
       'Cycle,Home,product_review,product_review,product_review3,ScanPrevi
ew,VerifyDateOfBirth,location,VerifyCountry,VerifyPhone,VerifyToken,produc
t_review,product_review,VerifySSN,product_review,SelectInstitution,BankVer
ification,product_review,product_review,',
       'product_review,ScanPreview,VerifyDateOfBirth,VerifyCountry,Profile
VerifySSN,ProfilePage,ProfileEducation,ProfileEducationMajor,Saving2Amoun
t,Saving8,ProfileMaritalStatus,ProfileChildren,Saving2,Saving9,Saving7,Sav
ing6,Saving5,Home,Loan2,'],
      dtype=object)
```

In [72]:

```
df5.columns
```

Out[72]:

```
Index(['user', 'dayofweek', 'hour', 'age', 'screen_list', 'numscreens',
       'minigame', 'used_premium_feature', 'enrolled', 'liked', 'screen_na
me'],
      dtype='object')
```

In [73]:

```
for screen_name in df_screen_list:
    df5[screen_name] = df5.screen_list.str.contains(screen_name).astype(int)
    df5['screen_list'] = df5.screen_list.str.replace(screen_name+",", "")
```

In [75]:

```
df5.head()
```

Out[75]:

	user	dayofweek	hour	age	screen_list	numscreens
0	235136	3	2	23	joinscreen,product_review,ScanPreview,VerifyTo...	15
1	333588	6	1	24	joinscreen,product_review,product_review2,Scan...	13
2	254414	1	19	23		3
3	234192	4	16	28	product_review,Home,product_review,ReferralCon...	40
4	51549	1	18	31	joinscreen,ScanPreview,VerifySSN,Home,SelectIn...	32

5 rows × 69 columns

In [77]:

```
df5.shape
```

Out[77]:

```
(50000, 69)
```

In [78]:

```
df5.loc[0,"screen_list"]
```

Out[78]:

```
'joinscreen,product_review,ScanPreview,VerifyToken,ProfileVerifySSN,Settin
gs,ForgotPassword,'
```

In [79]:

```
df5['remain_screen_list'] = df5.screen_list.str.count(",")
```

In [81]:

```
df5.drop(columns=["screen_list",'screen_name'],axis=0,inplace=True)
```

In [82]:

```
saving_screens = ['Saving1',
                  'Saving2',
                  'Saving2Amount',
                  'Saving4',
                  'Saving5',
                  'Saving6',
                  'Saving7',
                  'Saving8',
                  'Saving9',
                  'Saving10',
                 ]
df5['saving_screens_count'] = df5[saving_screens].sum(axis = 1)
df5.drop(columns = saving_screens, inplace = True)
```

In [83]:

```
credit_screens = ['Credit1',
                  'Credit2',
                  'Credit3',
                  'Credit3Container',
                  'Credit3Dashboard',
                 ]
df5['credit_screens_count'] = df5[credit_screens].sum(axis = 1)
df5.drop(columns = credit_screens, axis = 1, inplace = True)
```

In [86]:

```
cc_screens = ['CC1',
              'CC1Category',
              'CC3',
             ]
df5['cc_screens_count'] = df5[cc_screens].sum(axis = 1)
df5.drop(columns = cc_screens, inplace = True)
```

In [87]:

```
loan_screens = ['Loan',
                'Loan2',
                'Loan3',
                'Loan4',
               ]
df5['loan_screens_count'] = df5[loan_screens].sum(axis = 1)
df5.drop(columns = loan_screens, inplace = True)
```

In [89]:

```
df5.shape
```

Out[89]:

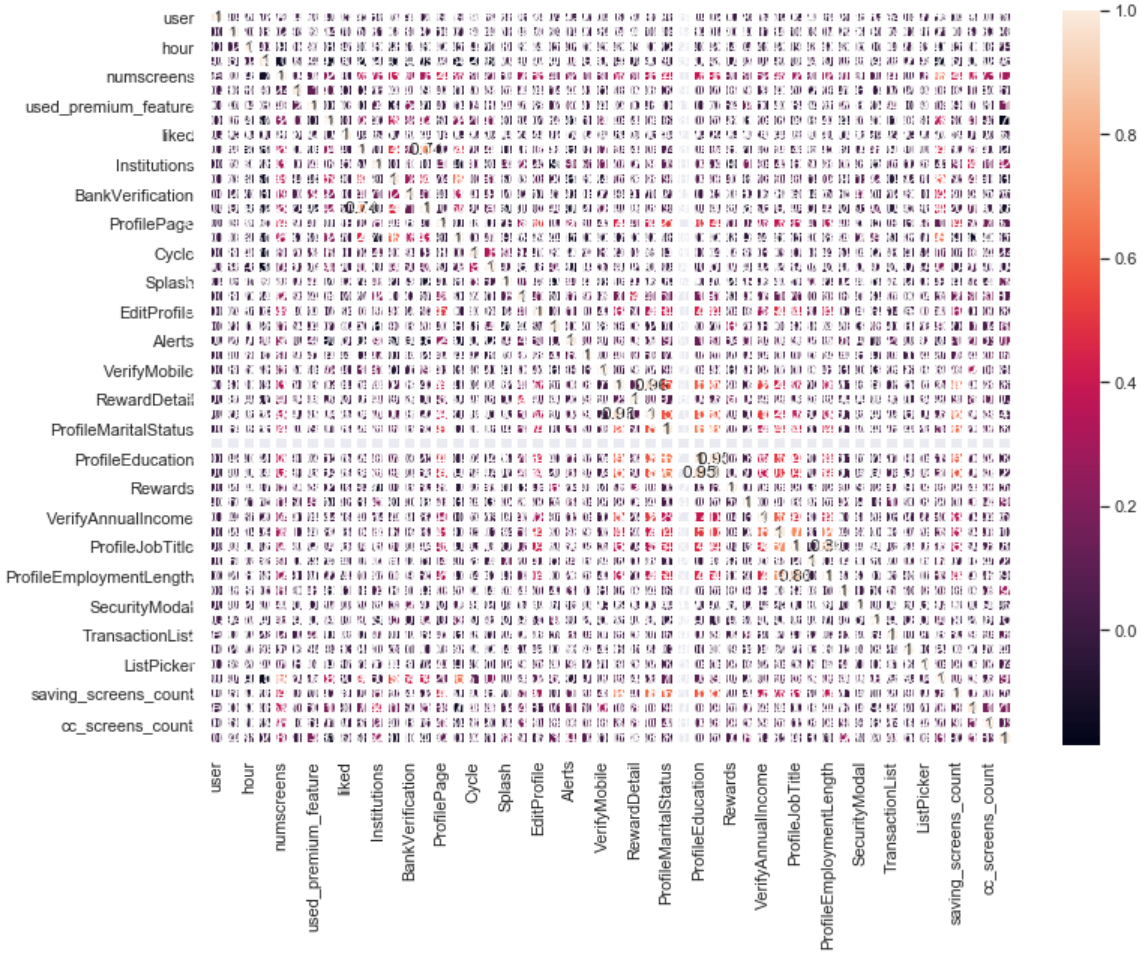
```
(50000, 50)
```

In [91]:

```
plt.figure(figsize=(12,9))
sns.heatmap(df5.corr(),annot=True,linewidth=3)
```

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bce347c388>
```



Data Preprocessing

In [92]:

```
Clean_fintech_data = df5.copy()
```

In [93]:

```
target = Clean_fintech_data["enrolled"]
fin_t_data = Clean_fintech_data.drop(columns= 'enrolled',axis=1)
```

In [94]:

```
target.count()
```

Out[94]:

```
50000
```

In [95]:

```
fin_t_data.shape
```

Out[95]:

```
(50000, 49)
```

In [96]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(fin_t_data,target,test_size=0.2,random
_state=0)
```

In [99]:

```
y_test.count()
```

Out[99]:

```
10000
```

In [100]:

```
# take User ID in another variable
train_userID = x_train['user']
x_train.drop(columns= 'user', inplace =True)
test_userID = x_test['user']
x_test.drop(columns= 'user', inplace =True)
```

```
C:\Users\shubh\anaconda3\lib\site-packages\pandas\core\frame.py:3997: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  errors=errors,
```

In [ ]:

```
Feature Scalling
```

In [104]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train_sc = sc.fit_transform(x_train)
x_test_sc = sc.transform(x_test)
```

In [105]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [106]:

```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(random_state = 0)
lr_model.fit(x_train, y_train)
y_pred_lr = lr_model.predict(x_test)

accuracy_score(y_test, y_pred_lr)
```

```
C:\Users\shubh\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[106]:

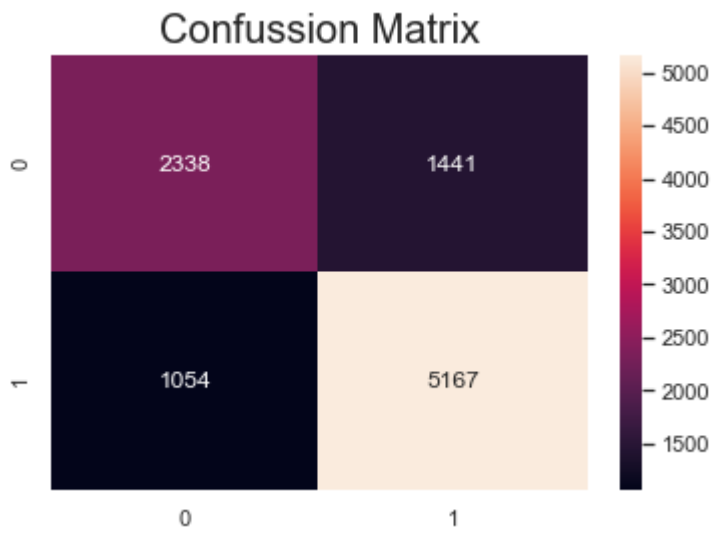
```
0.7505
```

In [107]:

```
cm_lr_pt2 = confusion_matrix(y_test, y_pred_lr)
sns.heatmap(cm_lr_pt2, annot = True, fmt = 'g')
plt.title("Confussion Matrix", fontsize = 20)
```

Out[107]:

```
Text(0.5, 1, 'Confussion Matrix')
```



In [108]:

```
final_result = pd.concat([test_userID, y_test], axis = 1)
```

In [109]:

```
final_result['predicted result'] = y_pred_lr

print(final_result)
```

```
         user  enrolled  predicted result
11841  239786         1                 1
19602  279644         1                 1
45519   98290         0                 0
25747  170150         1                 1
42642  237568         1                 1
...       ...       ...               ...
25091  143036         1                 0
27853   91158         1                 1
47278  248318         0                 0
37020  142418         1                 1
2217   279355         1                 1

[10000 rows x 3 columns]
```

In [ ]:

```

```