In [1]:
```
import pandas as pd
import altair as alt
import math
import scipy.stats
```

In [2]:
```
data = pd.read_csv("Group2_Dataset.csv")
```

In [3]:
```
data = data.drop(0)
```

Converting the data types from strings to numbers in the csv

In [4]:
```
#data["LightModeTime"] = pd.to_numeric(data["LightModeTime"])
#data["DarkModeTime"] = pd.to_numeric(data["DarkModeTime"])

columnNames = ["LightModeTime", "DarkModeTime", "LightModeAccuracy", "DarkModeAccuracy",

for colName in columnNames:
    data[colName] = pd.to_numeric(data[colName])
```

Dropped the first row as it has all of the question descriptions from Qualtrics

In [5]:
```
data["time_diff"] = data["DarkModeTime"] - data["LightModeTime"]
data["accuracy_diff"] = data["DarkModeAccuracy"] - data["LightModeAccuracy"]
data["comfort_diff"] = data["OverallDarkComfortScore"] - data["OverallLightComfortScore"
```

In [6]:
```
darkModeTimeSTD = data.DarkModeTime.std()
darkModeTimeMean = data.DarkModeTime.mean()

lightModeTimeSTD = data.LightModeTime.std()
lightModeTimeMean = data.LightModeTime.mean()

darkModeAccuracySTD = data.DarkModeAccuracy.std()
darkModeAccuracyMean = data.DarkModeAccuracy.mean()

lightModeAccuracySTD = data.LightModeAccuracy.std()
lightModeAccuracyMean = data.LightModeAccuracy.mean()

darkModeComfortSTD = data.OverallDarkComfortScore.std()
darkModeComfortMean = data.OverallDarkComfortScore.mean()

lightModeComfortSTD = data.OverallLightComfortScore.std()
lightModeComfortMean = data.OverallLightComfortScore.mean()
```

Checking for outliers (more than 3 standard deviations away from the mean):

In [7]:
```
data = data[(data["DarkModeTime"] <= (darkModeTimeMean + 3*darkModeTimeSTD))]
data = data[(data["DarkModeTime"] >= (darkModeTimeMean - 3*darkModeTimeSTD))]

data = data[(data["LightModeTime"] <= (lightModeTimeMean + 3*lightModeTimeSTD))]
data = data[(data["LightModeTime"] >= (lightModeTimeMean - 3*lightModeTimeSTD))]


data = data[(data["DarkModeAccuracy"] <= (darkModeAccuracyMean + 3*darkModeAccuracySTD))
data = data[(data["DarkModeAccuracy"] >= (darkModeAccuracyMean - 3*darkModeAccuracySTD))

data = data[(data["LightModeAccuracy"] <= (lightModeAccuracyMean + 3*lightModeAccuracyST
data = data[(data["LightModeAccuracy"] >= (lightModeAccuracyMean - 3*lightModeAccuracyST


data = data[(data["OverallDarkComfortScore"] <= (darkModeComfortMean + 3*darkModeComfort
data = data[(data["OverallDarkComfortScore"] >= (darkModeComfortMean - 3*darkModeComfort

data = data[(data["OverallLightComfortScore"] <= (lightModeComfortMean + 3*lightModeComf
data = data[(data["OverallLightComfortScore"] >= (lightModeComfortMean - 3*lightModeComf
```

In [8]:
```
data
```

Out[8]:

	ParticipantId	Experimenter	Informed Consent	LightModeTime	LightModeAccuracy	DarkModeTime	DarkMode
1	1	Team	Yes	76	0.96	75	
2	2	Team	Yes	62	0.97	60	
3	3	Team	Yes	77	0.96	72	
4	4	Team	Yes	76	0.97	98	
5	5	Team	Yes	72	0.96	77	
6	6	Team	Yes	86	0.96	88	
7	7	Team	Yes	56	0.97	54	
8	8	Team	Yes	53	0.95	53	
9	9	Team	Yes	68	0.96	79	
10	10	Team	Yes	71	0.96	73	
11	11	Team	Yes	57	0.94	74	
12	12	Team	Yes	43	0.97	46	
13	13	Team	Yes	46	0.93	41	
14	14	Team	Yes	66	0.88	67	
15	15	Team	Yes	44	0.89	56	
16	16	Oliver	Yes	79	0.94	72	
17	17	Lawrence	Yes	67	0.97	71	
18	18	Oliver	Yes	78	0.92	80	
19	19	Oliver	Yes	61	0.95	55	
20	20	Oliver	Yes	57	0.96	56	
21	21	Oliver	Yes	88	0.97	92	
22	22	Lawrence	Yes	97	0.93	107	
23	23	Lawrence	Yes	45	0.93	48	
24	24	Lawrence	Yes	72	0.85	70	
25	25	Lawrence	Yes	67	0.89	61	
26	26	Kevin	Yes	97	0.97	102	

28	28	Kevin	Yes	80	0.98	72
29	29	Kevin	Yes	65	0.96	69
30	30	Kevin	Yes	72	0.94	67

In [55]:
```
dfMeltTime = pd.melt(data, id_vars="ParticipantId", value_vars=["LightModeTime", "DarkMo
dfMeltTime.columns = ["ParticipantId", "Theme", "Time"]

dfMeltAccuracy = pd.melt(data, id_vars="ParticipantId", value_vars=["LightModeAccuracy",
dfMeltAccuracy.columns = ["ParticipantId", "Theme", "Accuracy"]

dfMeltComfort = pd.melt(data, id_vars="ParticipantId", value_vars=["OverallLightComfortS
dfMeltComfort.columns = ["ParticipantId", "Theme", "Comfort"]
```

Changing the dataframe for visualization purposes

Hypothesis 1: Using Dark Mode will result in lower task completion times

$H_0 : \mu_D = \mu_L$

$H_A : \mu_D < \mu_L$

Assumption Checks

In [9]:
```
scipy.stats.shapiro(data.time_diff)
```

Out[9]:
```
ShapiroResult(statistic=0.9228260517120361, pvalue=0.035977207124233246)
```

Assumption was not met because we got a p-value of 0.036 which is less than our the overall α of 0.05. The histogram below also supports the notion that we didn't meet the assumption since the histogram isn't normally distributed.

In [10]:
```
alt.Chart(data, title="Histogram of change in time of completion").mark_bar().encode(
    x=alt.X("time_diff", bin=True, title="Change in time completion"),
    y="count()"
)
```

Out[10]:

Descriptive Statistics

In [11]:
```
darkModeTimeSTD = data.DarkModeTime.std()
darkModeTimeMean = data.DarkModeTime.mean()
```

```
lightModeTimeSTD = data.LightModeTime.std()
lightModeTimeMean = data.LightModeTime.mean()

darkModeTimeMedian = data.DarkModeTime.median()
lightModeTimeMedian = data.LightModeTime.median()

print("Dark: mean = %.2f, median = %.2f, SD = %.2f" % (darkModeTimeMean, darkModeTimeMed
print("Light: mean = %.2f, median = %.2f, SD = %.2f" % (lightModeTimeMean, lightModeTime
```

```
Dark: mean = 70.17, median = 71.00, SD = 16.41
Light: mean = 68.21, median = 68.00, SD = 14.59
```

We denote the median because of the nature of non-parametric tests and that they choose to focus on the median. We also need to adjust our hypothesis to reflect this change.

New Hypothesis:

$H_0 : med_D = med_L$

$H_A : med_D < med_L$

Wilcoxon T-test

Since the Shapiro-Wilk test failed and we didn't get a p-value greater than 0.05, we have to proceed with the Wilcoxon Test, which is a non-parametric test for a within-subjects design with 2 conditions, and this lines up with our experimental design.

```
In [12]: scipy.stats.wilcoxon(data.DarkModeTime, data.LightModeTime, alternative='less')
```

```
Out[12]: WilcoxonResult(statistic=245.5, pvalue=0.8339345454738164)
```

Since we have an overall α of 0.05 and we are running 3 total tests, we need to correct for multiple comparisons by doing Bonferroni's correction to find an adjusted α of $0.05/3$. The p-value of 0.834 is above our corrected α of 0.017, so we do not have statistically significant results.

Effect Size

```
In [13]: cohen_d_time = data.time_diff.mean() / data.time_diff.std()
         cohen_d_time
```

```
Out[13]: 0.27513446682832354
```

This means that there is only a small effect because the value of 0.275 is in the range between 0.2 and 0.5 and this is classified as the small range

Confidence Intervals

For a non-parametric test, like the Wilcoxon test, we cannot confidently say that there is a confidence interval that represents the true mean difference in typing completion times because non-parametric tests focus on the median, not the mean.

Data Visualization

In [36]:
```
alt.Chart(dfMeltTime).mark_boxplot(extent='min-max').encode(
    x='Time',
    y='Theme'
)
```

Out[36]:

Hypothesis 2: Using Dark Mode will result in higher accuracy scores

$H_0 : \mu_{AD} = \mu_{AL}$

$H_A : \mu_{AD} > \mu_{AL}$

Assumption Checks

In [48]:
```
scipy.stats.shapiro(data.accuracy_diff)
```

Out[48]:
```
ShapiroResult(statistic=0.9397556185722351, pvalue=0.09880173206329346)
```

Assumption was met because we got a p-value of 0.099 which is greater than the overall α of 0.05. The histogram below is also normally distributed.

In [49]:
```
alt.Chart(data, title="Histogram of change in typing accuracy").mark_bar().encode(
    x=alt.X("accuracy_diff", bin=True, title="Change in typing accuracy"),
    y="count()"
)
```

Out[49]:

Descriptive Statistics

In [20]:
```
darkModeAccuracySTD = data.DarkModeAccuracy.std()
darkModeAccuracyMean = data.DarkModeAccuracy.mean()

lightModeAccuracySTD = data.LightModeAccuracy.std()
lightModeAccuracyMean = data.LightModeAccuracy.mean()

print("Dark: mean = %.2f, SD = %.2f" % (darkModeAccuracyMean, darkModeAccuracySTD))
print("Light: mean = %.2f, SD = %.2f" % (lightModeAccuracyMean, lightModeAccuracySTD))
```

```
Dark: mean = 0.94, SD = 0.03
Light: mean = 0.94, SD = 0.03
```

One-Tailed Paired Sample T-Test

In [21]:
```
scipy.stats.ttest_rel(data.DarkModeAccuracy, data.LightModeAccuracy, alternative='greate
```

Out[21]:
```
Ttest_relResult(statistic=-0.07304358872384731, pvalue=0.5288546328856495)
```

The p-value of 0.529 is greater than our adjusted α of 0.017, so we do not have statistically significant results and we fail to reject the null hypothesis that there is no difference in the mean accuracy scores between light and dark mode.

Effect Size

In [22]:
```
cohen_d_accuracy = data.accuracy_diff.mean() / data.accuracy_diff.std()
cohen_d_accuracy
```

Out[22]:
```
-0.013563853909740315
```

This means there is only a small effect on accuracy because the magnitude of Cohen's d for the accuracy is 0.014 which is in the small range.

Confidence Intervals

In [23]:
```
bounds_accuracy = scipy.stats.norm.interval(alpha=0.95, loc=data.accuracy_diff.mean(), s
bounds_accuracy
```

Out[23]:
```
(-0.009597517132945267, 0.008907861960531475)
```

This tells us that 95% of the time, the true mean difference in accuracy scores for both light and dark mode will fall between -0.0096 and 0.0089 units. This lines up with the fact that we didn't reject the null hypothesis (since we didn't get a statistically significant result) because 0 is within this range, meaning there could be no difference in dark and light mode accuracy scores. If there's no difference, then the mean accuracy for dark mode isn't greater than the mean accuracy for light mode.

In [93]:
```
n1 = len(data.DarkModeAccuracy)
n2 = len(data.LightModeAccuracy)
darkAccuracySE = (darkModeAccuracySTD / math.sqrt(n1))
lightAccuracySE = (lightModeAccuracySTD / math.sqrt(n2))

bounds_darkAccuracy = scipy.stats.norm.interval(alpha=0.95, loc=darkModeAccuracyMean, sc

bounds_lightAccuracy = scipy.stats.norm.interval(alpha=0.95, loc=lightModeAccuracyMean,

print("Confidence Interval for Dark Mode Accuracy: ", bounds_darkAccuracy)
print("Confidence Interval for Light Mode Accuracy: ", bounds_lightAccuracy)
```

```
Confidence Interval for Dark Mode Accuracy:  (0.9160515559730846, 0.9559484440269153)
Confidence Interval for Light Mode Accuracy:  (0.9224537862501285, 0.9548795470832048)
```

This tells us that 95% of the time, the true value of the population mean of the accuracy for Dark Mode will fall between 0.916 and 0.956. Additionally, the true value of the population mean of the accuracy for Light Mode will fall between 0.922 and 0.955. These intervals are extremely similar and this supports the fact that we fail to reject the null hypothesis that the accuracy for both light and dark mode are equal.

Data Visualization

In [95]:
```
intervalsAccuracy = pd.DataFrame({'Theme': ["Light","Dark"],
                                'Mean':  [lightModeAccuracyMean, darkModeAccuracyMean],
                                'lower': [bounds_lightAccuracy[0], bounds_darkAccuracy[0]],
                                'upper': [bounds_lightAccuracy[1], bounds_darkAccuracy[1]]})
```

In [96]:
```
pts = alt.Chart(intervalsAccuracy).mark_point().encode(
    x=alt.X("Mean", scale=alt.Scale(domain=[0.9, 1.0])),
    y="Theme"
).properties(width=200)

error_bars = alt.Chart(intervalsAccuracy,title="Confidence Intervals for Accuracy by The
    y=alt.Y("Theme",title="Theme"),
    x=alt.X("lower", title="Accuracy", scale=alt.Scale(domain=[0.9, 1.0])),
    x2="upper"
)

pts+error_bars
```

Out[96]:

Hypothesis 3: Dark Mode will have a higher perceived comfort score

$H_0 : \mu_{CD} = \mu_{CL}$

$H_A : \mu_{CD} > \mu_{CL}$

In [26]:
```
scipy.stats.shapiro(data.comfort_diff)
```

Out[26]:
```
ShapiroResult(statistic=0.9796977639198303, pvalue=0.8304286599159241)
```

Assumption was met because we got a p-value of 0.83 which is greater than the overall α of 0.05. The histogram below is also normally distributed.

In [27]:
```
alt.Chart(data, title="Histogram of change in perceived comfort score").mark_bar().encod
    x=alt.X("comfort_diff", bin=True, title="Change in perceived comfort score"),
    y="count()"
)
```

Out[27]:

Descriptive Statistics

In [43]:
```
darkModeComfortSTD = data.OverallDarkComfortScore.std()
darkModeComfortMean = data.OverallDarkComfortScore.mean()

lightModeComfortSTD = data.OverallLightComfortScore.std()
lightModeComfortMean = data.OverallLightComfortScore.mean()

print("Dark: mean = %.2f, SD = %.2f" % (darkModeComfortMean, darkModeComfortSTD))
print("Light: mean = %.2f, SD = %.2f" % (lightModeComfortMean, lightModeComfortSTD))
```

```
Dark: mean = 11.97, SD = 4.14
Light: mean = 10.83, SD = 3.61
```

One-Tailed Paired Sample T-Test

In [44]:
```
scipy.stats.ttest_rel(data.OverallDarkComfortScore, data.OverallLightComfortScore, alter
```
Out[44]:
```
Ttest_relResult(statistic=0.8853336897019269, pvalue=0.19175846073217295)
```

The p-value of 0.19 is greater than our adjusted α of 0.017, so we do not have statistically significant results and we fail to reject the null hypothesis that there is no difference in the mean perceived comfort scores between light and dark mode.

Effect Size

In [45]:
```
cohen_d_comfort = data.comfort_diff.mean() / data.comfort_diff.std()
cohen_d_comfort
```
Out[45]:
```
0.1644023389087364
```

This means there is only a small effect on perceived comfort score because the magnitude of Cohen's d for the perceived comfort score is 0.1644 which is in the small range.

Confidence Intervals

In [46]:
```
bounds_comfort = scipy.stats.norm.interval(alpha=0.95, loc=data.comfort_diff.mean(), sca
bounds_comfort
```
Out[46]:
```
(-1.3812364505222565, 3.657098519487774)
```

This tells us that 95% of the time, the true mean difference in the mean comfort scores between light and dark mode will fall between -1.38 and 3.66 units. This lines up with the fact that we didn't reject the null hypothesis (since we didn't get a statistically significant result) because 0 is within this range, meaning there could be no difference in dark and light mode perceived scores. If there's no difference, then the mean perceived comfort score for dark mode isn't greater than the mean perceived comfort score for light mode.

In [94]:
```
n3 = len(data.OverallDarkComfortScore)
n4 = len(data.OverallLightComfortScore)
darkComfortSE = (darkModeComfortSTD / math.sqrt(n3))
lightComfortSE = (lightModeComfortSTD / math.sqrt(n4))

bounds_darkComfort = scipy.stats.norm.interval(alpha=0.95, loc=darkModeComfortMean, scal

bounds_lightComfort = scipy.stats.norm.interval(alpha=0.95, loc=lightModeComfortMean, sc

print("Confidence Interval for Dark Mode Perceived Comfort Scores: ", bounds_darkComfort
print("Confidence Interval for Light Mode Perceived Comfort Scores: ", bounds_lightComfo
```
```
Confidence Interval for Dark Mode Perceived Comfort Scores:  (10.616857637981669, 13.649
809028684997)
Confidence Interval for Light Mode Perceived Comfort Scores:  (9.28676898035789, 11.9798
97686308776)
```

This tells us that 95% of the time, the true value of the population mean of the perceived comfort score for Dark Mode will fall between 10.617 and 13.650. Additionally, the true value of the population mean of the perceived comfort score for Light Mode will fall between 9.287 and 11.980. These intervals are similar and this supports the fact that we fail to reject the null hypothesis that the perceived comfort scores for both light and dark mode are equal.

Data Visualization

In [98]:
```
intervalsComfort = pd.DataFrame({'Theme': ["Light","Dark"],
                                 'Mean':  [lightModeComfortMean, darkModeComfortMean],
                                 'lower': [bounds_lightComfort[0], bounds_darkComfort[0]],
                                 'upper': [bounds_lightComfort[1], bounds_darkComfort[1]]})
```

In [100…
```
pts = alt.Chart(intervalsComfort).mark_point().encode(
    x=alt.X("Mean", scale=alt.Scale(domain=[8, 15])),
    y="Theme"
).properties(width=200)

error_bars = alt.Chart(intervalsComfort,title="Confidence Intervals for Perceived Comfor
    y=alt.Y("Theme",title="Theme"),
    x=alt.X("lower", title="Perceived Comfort Scores", scale=alt.Scale(domain=[8, 15])),
    x2="upper"
)

pts+error_bars
```

Out[100]:

In [ ]: