VISION STATEMENT

PRODUCT NAME: MAGIC MINUTES
TEAM NAME: AND YET… IT COMPILES
SLACK: HTTPS://ANDYETITCOMPILESGROUP.SLACK.COM/MESSAGES/CHNK9FN07/
GitHub: HTTPS://GITHUB.COM/COLEBERGMANN/AND-YET-IT-COMPILES
Trello: https://trello.com/b/iG3UuHoD/cs48

TEAM PERCEPTION

<table>
<thead>
<tr>
<th>Position</th>
<th>Name</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Team Lead</td>
<td>Abby Wysopal</td>
<td><a href="mailto:abbywysopal@gmail.com">abbywysopal@gmail.com</a></td>
</tr>
<tr>
<td>Developer</td>
<td>Olivia Clough</td>
<td><a href="mailto:olivia_clough@ucsb.edu">olivia_clough@ucsb.edu</a></td>
</tr>
<tr>
<td>Developer</td>
<td>Denver Simmons</td>
<td><a href="mailto:denverjamessimmons@gmail.com">denverjamessimmons@gmail.com</a></td>
</tr>
<tr>
<td>Developer</td>
<td>Cole Bergmann</td>
<td><a href="mailto:colebergmann@ucsb.edu">colebergmann@ucsb.edu</a></td>
</tr>
<tr>
<td>Developer</td>
<td>Ryan Mitchell</td>
<td><a href="mailto:ryanm777@gmail.com">ryanm777@gmail.com</a></td>
</tr>
</tbody>
</table>

BACKGROUND

Disneyland Parks in Anaheim, California attracts over 44,000 visitors per day. Each visitor arrives at Disneyland with the risk of waiting in line for multiple hours per attraction. Unnecessary stress is imposed upon visitors when waiting in excruciatingly long lines and being surrounded by thousands of people. We believe there should be a way to predict wait times and crowd sizes within each park. Through the prediction of wait times per ride at Disneyland, visitors will enjoy a more efficient day at Disneyland parks. Thus, making the happiness place on earth even happier.

Today, Disneyland currently has a mobile application that contains a variety of features. Disney, however, is a company focused mainly on increasing its profit margins, and therefore the user interface of their application focuses on advertising ticket sales, dining features, and fast passes in the ride time section. Disney has all of the necessary data to predict wait times, but they do not give consumers access to this information because it could negatively impact ticket sales. We hope to avoid this and share accurate predictions.

With their application, they only provide live updates. Therefore, the general public cannot plan their trips or visits accordingly. In our service, we will use machine learning to train an algorithm using data points found from their live updates. We will push ourselves to learn more technologies in order to solve a problem that we find interesting.
PROJECT

Outcome
The outcome of this project will be an application that allows the user to harness our trained models and predict the best day to attend Disneyland based on average wait times and park population density. The user will also be able to use daily predictions to plan when to go on individual rides in the park.

Milestones
The most essential component of this project is creating an accurate machine learning model to predict future wait times for Disneyland rides. Firstly, we will gather relevant data such as weather, wait times and population at each park. Creating a set of data which will be formatted to train our model. We will then research different machine learning frameworks in order to further train our model. As we train the model, we will create the web interface use ReactJS. After selecting a time and date, a graph of predicted wait times will be generated and shown to the user. While our backend data will be handled by either AWS/Google Firebase.

Sprint 1 (4/15 - 4/26)

a. Gather relevant data to feed the neural network
b. Format the data for the network to read
c. Determine a machine learning framework/service to use
d. Determine the most effective network structure / hyperparameters for our problem
e. Begin training the machine learning algorithm with our dataset
f. Create a basic web interface with React JS

Sprint 2 (4/29 - 5/10)

g. Explore different databases (cloud services, local sql, etc) and pick a solution
h. Update web interface to link between pages
i. Use Node JS for backend development and API call
j. Type in a ride, return a google maps type graph of what the best time to go on it is
k. Display a google maps type graph for park crowds for the next 30 days

Sprint 3 (5/13 - 5/24)

l. Finalize the web interface
m. Update models for increased accuracy
n. Input real time data into ML model
o. Clean up the web app backend in Node JS

Optional (if time permits)

p. Add the capability for the user to set a date and select a list of rides they want to go on. The app will create the best schedule.

q. Further improve the model with additional data points if we have time (% of schools in session, etc)

r. Further improve the front end of our website application, and possibly adapt our website application to iOS and Android.
High Level Overview
Detailed Design
UML Diagrams

1) Machine Learning
   - MyModels
     - Operations:
       - LoadCSV(csv_file)
       - Predict(model)
   - CSV File
   - Model
     - Attributes:
       - Machine Learning Model
     - Operations:
       - LoadModel(model)
       - Predict()

2) Live Data
   - Live Data
   - LocalWeather
     - dayHigh
     - dayLow
     - dayPrecipProb
     - tempNow
     - precipProbNow
     - Constructor
     + getDayHigh
     + getDayLow
     + getDayPrecipProb
     + getTempNow
     + precipProbNow
     - getWeatherNow
     - streamToJson
   - ParkHours
     - dOpenHour
     - dCloseHour
     - dcaOpenHour
     - dcaCloseHour
     + Constructor
     + getDOpenHour
     + getDCloseHour
     + getDcaOpenHour
     + getDcaCloseHour
     - waitForTime
     - fetchCurrentTime
     - formatDate
   - CurrentWaitTimes
     - rideWaits[]
     + Constructor
     + getWaitTime
     + isDown
     + isClosed
     + fetchLatestWaits
     - parseRide
3) UI Today Page

4) UI Plan Page
Sequence Diagrams

1) Interaction between Classes

![Diagram 1)
Interaction between Classes]

1.1 Import PlanGraph

Class: Plan

1.2 getCharData()
API Call

1.3 Return data array

Class: PlanGraph

1.4 Render() graph

Class: Database

2) Interaction with User and System

![Diagram 2)
Interaction with User and System]

User

1. Open Web App

Web App

1.1 Display Home Page

2. Select Today Link

2.1 Display Today Page

3. Select Ride

3.5 Display Ride Wait time graph

API

3.1 WaitTimeData

3.2 Real Time & Predictive Data

3.4 Array of Wait time Data

ML

3.3 Return Data
UI Mockups

Welcome to Magic Minutes

Disneyland Park

Population

Months

Park crowds for the upcoming month:

Population

Day

Please choose ride:

Ride Name

Wait Time

Hour
Feature Cases: User stories

Web App:

1. Annual Population
As a web page visitor, I want to see which month out of the year is the most crowded.
Scenario 1: Annual Population
Given I am a user
And I want to see the different month’s population Disney
When I navigate to the home page
Then I will see a graph comparing the crowds for each month of the year

2. Homepage Options
As a web page visitor, I want to select a time range in order to see graphed data of wait times or population at Disneyland
Scenario 1: Web Page formatting
Given I am on the website front page
And there is a selection of today or planner
When I select today
Then I will be directed to a new web page to choose my ride
And when I select planner
Then I will be directed to a new web page to help plan my Disneyland trip

3. Ride and Times
As a Disneyland Park visitor, I can see the ride wait times for today so that I can plan the best time to go on the ride.
Scenario 1: I am in Disney today only.
Given the date is today,
And the ride is Space Mountain.
When the user chooses the ride
Then the graph will show me the wait times,
And I will be able to pick a time to go.

4. Date Range and Crowds
As a User, I can choose to plan my trip so that I can see what days in the next month are the busiest.
Scenario 1: I want to go to Disney in the next month.
Given the user chooses planning page
and the range is at most 30 days,
When the user clicks Search
Then a graph of the park population for the next month will appear.

5. Web App Backend to access ML model
As a website, I can connect to a server and access the ML model so that I can display the results of the model.
Scenario 1: Website calls model
Given the website calls the model,
and the model is hosted correctly,
The model can take inputs from the website
and provide the website with the model’s output.
ML Network:

6. **Transform data into segments**
As a machine learning bot, I have the relevant historical ride data in the format I need.
Scenario 1:
Given a database of historical ride data,
When the Java program is run,
Then a csv file with all the historical data points for Disney rides are outputted
And they are in the correct format to be fed into the model training.

7. **Trained Neural Network**
As a programmer, I can implement the trained neural network into my software so that it actively predicts the wait times of a ride over the current day when fed park data in real time.
Scenario 1: I am programming software that uses predictive modeling
Given that I have real time disneyland historical data
And I input the real time data into the network
The network outputs ride wait predictions for the day

8. **Collect Data Points**
As a programmer, I can gather valid points of data so that the TensorFlow model can run accurately.
Scenario 1: API setup is correct
Given the data collection is exhausted
And the programmer has formatted it correctly
When the ML TensorFlow is ready
Denver and Ryan will be able to input an API

9. **Connection between back and front end**
As a web page interface, I will be able to call python scripts to display the ML model data.
Scenario 1: HandleSubmit
Given I am a web page
And I want to display the model information
When a button is pressed, I will call the python script
Then I be given an array style API of predictive data

**Sprint Progress**

At the end of sprint 1, we have a working ML model for one ride. This is set up in a way to be able to change one number in order to change the ride (CSV is formatted correctly for this as well). We also have a running UI with all components that is hosted locally.

In sprint 2, we are focusing on getting the model up and running for all 10 rides as well as the population of the next 30 days. Connecting the two parts of the project will bring us to a place to be ready for sprint 3.