The approach I used for this project entails the following sections:
-Vehicle
-Behavior
-Trajectory
-Path
-Input to the simulator

-VEHICLE:
This class contains the information about the vehicle. In particular, its position on the road (defined by Frenet coordinates "s" and "d"), its velocity and the lane it occupies on the road, together with the knowledge of its surrounding lanes.
I have used this class to keep information about my vehicle and the other vehicles on the road.
Furthermore, I saved its position and velocity (on the "s" and "d" coordinates) to be used as a starting point for each iteration.

-BEHAVIOR:
The behavior section takes as inputs the information about my car and the other vehicles. Based on those, it outputs the behavior that my car should take in the form of "KEEP LANE", "TURN RIGHT", "TURN LEFT".
In details, I used several methods to calculate, for each lane, the gap between my vehicle and the other vehicles in front and behind. For each situation I assigned a cost function that would penalize or reward positions that are within a certain threshold in regards to the vehicle in front or behind.
I used a smaller threshold for the gap behind the vehicle to allow a bit more aggressive behavior.
The use of the cost function allows to discern the positions that are "safe" to discover and therefore to provide an answer in regards to which action to take.

-TRAJECTORY:
The trajectory function takes as inputs the information about my vehicle and the behavior that should be taken.
The result is the target position on the road (in terms of Frenet coordinates "S" and "d").
The trajectory function that I used considers a "transfer time" of 2 seconds, which would take the car for moving from the current position to the target location.
The target speed is evaluated in respect to the vehicle's speed, the speed limit and the front vehicle's speed.
In details, I set the "s" and "d" acceleration components for the starting and ending state to zero. Also the velocity component of the "d" coordinate in the final state is zero as I don't expect the car to have a lateral velocity component once the final position is reached.
As I mentioned the trajectory is calculate for both coordinates "s" and "d".
The result of this function is passed onto the JMT (Jerk Minimizing Trajectory) function.
"Jerk" is defined as the instantaneous change in acceleration.

The inputs of the JMT function are the starting and ending status of the vehicle (in terms of location, velocity and acceleration) and the “transfer time”.
The JMT provides us a position function (through a quintic polynomial solver) that minimizes the squared jerk value.
Once again, JMT is calculated for both coordinates, therefore taking in consideration the lateral and longitudinal jerks.

-PATH:
The Path module takes the result of the above and spits out a path in terms of X and Y coordinates, when given the time increment between the points and the number of points we want to use.
The function “convert_sd_to_XY” uses a spline function to estimate a cubic polynomial that encompasses all the waypoints along the road and then finally it converts the calculated ”s” and “d” points into discrete X, Y points.
(In this project the time increment is 0.02 seconds and the total number of points I used is calculated in respect to the “transfer time” and the time increment).

-INPUT TO THE SIMULATOR:
The final step is to put all these sections together and feed the discrete X, Y points to the simulator.

At the beginning of the track I did not consider the other vehicles for the first few seconds.
I experimented several values using initially only 10 points then increasing them by hundreds in order to reduce the “jerk” on the acceleration.
I settled with 225 points, which allow the car to reach the target speed with minimal jerk acceleration.
After this initial stage I used the pipeline described above to calculate the new waypoints to be passed to the simulator.