Terry:
Hey everybody, and welcome to our final project presentation for SP studios for our ParkInLot app. So first things first, we're going to go over some introductions, each of us are going to introduce ourselves, just explain what we did for the project and what experience we brought in to the project. My name is Terry Watson. I'm currently a CSUB. Student. My experience that I brought to the project was in SQL, HTML and PHP. The latter two I mostly learned during the project in terms of what I did for features for the project, I worked a lot on the database implementations, specifically with triggers for various features within the site. And then I also implemented reCAPTCHA, for the front end for various different sites security.

David:
I'm David Montes De Oca. Coming into this project, I had some experience with transact SQL, MySQL, basic web development, and some PHP. With this project, I mainly worked on the back end and the database, as well as the algorithm we use for the parking spot trading.

Abraham:
My name is Abraham, I have experience with HTML, CSS, JavaScript, some UI and UX design as well. And I was responsible for the front end and the design of the app.

Tony:
Hi, everyone. I'm Tony Cervantes. For this project. I'm the person that was mostly in charge of general project overview. So making sure everyone was working on what they were supposed to, and essentially pushing their updates to our GitHub repository. Thankfully, I've already had experience with most of the things we were going to be using, thanks to our web development two course. And that included things like PHP, Maria database, or MySQL, as well as general HTML and JavaScript coding. So for the idea of this project, I came up with this project, essentially, about three to four semesters ago, one of my classes would begin around 11am; So I would get to campus by 10:40. And one of these days that I was looking for a parking spot, I found a yellow vehicle, kind of like a dune buggy or a little Jeep. And essentially, I was able to take that person's spot for that day. Thanks to our schedules usually being like monday, wednesday or Tuesday, Thursdays, it seemed that every time my class that started at 11 was the same time when I assume his class ended around 10:30. Or he just decided to leave every single time around the same time, I would just wait right behind his vehicle until the vehicles owner came and drove off and left. So essentially, I just started taking his spot every single time my class was gonna start. And this worked out throughout essentially the whole semester. So this started making me think if there was already an application that exists that facilitates this trade off, where it's peer to peer or anything similar. It doesn't seem like there was so this is where our project began.

Terry:
Hey, it's me, again, we're gonna go over what's the problem that we're trying to address now. So here we have a map of CSUB. All these gray boxes are the various different parking lots that are available to students and faculty on campus. As you can see, there's quite a few different parking lots. So really, there should be no issue with people having enough space. But majority of the reason why there's an issue with parking is that whenever somebody is leaving, it's it's always difficult for somebody who's trying to arrive to be in the right place at the right time to get that spot. We as CSUB students and faculty all have a pretty bad time trying to park at the university. This is especially true during peak times. And also peak times of the year like final weeks and stuff like that. Whenever there's a lot of people on campus trying to park and leave simultaneously, it always creates a lot of chaos with a lot of people looping around through the parking lots, creating lots of traffic, lots of stress.

David:
To solve the problem that Terry described earlier, we created the application ParkInLot . So ParkInLot is essentially a tool that allows users to trade parking spots with one another. This uses a token based system just so we can ensure that people do not abuse the app. And ultimately, it just helps save stress and time.

So users will start off by registering for an account. And once they log in, they will have the option to either find a spot or give out a spot and when a user offers up a spot, their spot will be entered into our database and it will be ready to be paired with a requester. And once a requester is found, they will be shown the details as you can see there of the requesters car just to make the transaction a bit easier. Users who request a spot will be entered into our spot queuing algorithm. And once a match is found, they will be also given details for the trade. The users will also be given a complete button just to indicate when the spot trade is complete. And once this trade is complete, tokens will be exchanged, so the offer will gain a token and the requester will spend one of their tokens and ultimately these tokens will help keep the system fair and ensure that these users do not abuse the app, and they will need to offer up spots in order to be able to request spots in the future. Lastly, we implemented a map that automatically zooms in as you drive closer to the spot. Generally, this will be more useful once you're near campus and can actually see the parking spots clearly. This makes it easier for the requester to actually find the parking spot and will help us provide users with a better experience.

Terry:
But now I'm going to talk about our competition. In terms of direct competition, we don't have any. But there are some similar applications out there. These are pavement, Park, mobile and park stash. All three of these apps are much more similar to renting a parking space from its direct owner and things like Park driveways, and private parking lots. The problem with this is it can't be adapted to any parking lot unless monitoring

systems are in place. So things like cameras or meters so that the people who own the parking spaces are actually able to judge whether or not somebody is there. Another important distinction is that our application is free to use. And it's much more geared toward trading and peer to peer use.

Now I'm gonna go over the management style that we used. We used a combination of extreme programming and Scrum, both of which are agile methods. The reason why we switched over to Scrum halfway through our project is because this is all been online. And the scrum format is much more usable online, than extreme programming, which requires like pair programming and things like that. By using Scrum, we were able to just meet up whenever times called for it. And we could get a lot of work done that way individually.

Tony:
Okay, so now let's talk about our overall architecture. Essentially, our whole web application's foundation is coded in HTML and JavaScript, with CSS handling all of the styling. We used Odin's Maria DB to host all of our content's tables, which is from users to spots as well as using Odin itself to host our site. In order to access content from our database and display it onto their appropriate pages, PHP handles most of the static content. That which appears upon accessing a specific page that isn't going to be changed in real time immediately. For our algorithm, which is where we do need content to be updated almost immediately. And without having the page as a whole refresh, we used Ajax to handle the real time content acquisition. In terms of our external resources. We had Imgur's API, which allowed us to save our users vehicle pictures onto their own servers. And that way, the only thing being saved within our server is the link to those external images, as well as two different Google API's, which would be for their maps as well as reCAPTCHA.

Terry:
Now I'm going to go over the work plan that we had for semester one. In semester one, we focus mainly on the foundation and the groundwork for setting up the project and getting it operational. For about three to four weeks, we spent that creating the database for the users profile, house it attributes. This is all located in the users table on the database. For about three to five weeks, we moved on to creating a basic web view for signing in and signing up. For the last three weeks, we focused on implementing more options for the front end view. This includes adding more admin views. So being able to access database information as an admin.

Now I'm gonna move on to the work plan for semester two, semester two was definitely a lot higher workload than semester one or the first three to four weeks, we focused on getting the spot queue database tables operational. This includes things like the actual spots table itself, and all of the necessary attributes added to that table to facilitate spot trades. For four weeks after that, we focused on getting the front end view for spot queuing and trading actually working that was handled almost exclusively by David for two to three weeks that we focused on getting the token system properly working and other various triggers working on the database that was all handled by me, Terry. For two weeks after that, it was primarily focused on getting alternative views for the different roles. This includes employees, users and administrators that was all handled by Tony for the last three to four weeks, we focus really heavily on the front end view for Google Maps. This allows users to see the location of the trade partners. And then we also focused really heavily on various chase goals. This is primarily styling and in improving the readability of the website.

David:
So now we're moving on to the features that we all completed individually. I first focused on implementing photo upload so that users can add a picture of their vehicle to their profiles, and I used the Imgur API to store and fetch these uploads pretty easily. I also implemented a password hashing and sanitation feature to securely store users passwords and other input data using the password hash function in PHP as well as the bind param function. So these two features are really important just to help protect our database from potential SQL injection as well as protecting the user's information. The next feature I completed was the parking spot requests. Users are able to request parking spots be paired with an offeror, and then there'll be provided with details on the parking spot in order to complete the trade. I implemented this feature along with a few others using Ajax, PHP and JavaScript. So similar to the request feature, the offer feature will give users the ability to offer their spots be paired with the requester and be provided with the details in order to complete the spot trade. And lastly, I implemented a spot queuing algorithm just to ensure that the requesters and offers are matched based on availability, location and time. To do this, I used Ajax calls to continuously get updates from our database. This then allows us to update and display the parking spot's availability and status accordingly. Both users are shown updates once the status of the parking spot changes, just to help make the transaction simple and straightforward.

Terry:
Now moving on to the features I completed. throughout the project, I mainly worked on three things the reCAPTCHA for the site and also a lot of the database implementation. When I chose the reCAPTCHA, it was out of two potential features we were considering to implement for added security. They were two factor authentication and reCAPTCHA. I eventually decided on reCAPTCHA because while our site does have user profiles, and we want them to be secure, it doesn't necessarily have any super important user information. The most personal information our website houses for users as their car info down to the last four digits of their license plates and their names. Rather than implement two factor authentification to improve account security, I decided on reCAPTCHA to improve database slash site resource security. I implemented the captchas on three different pages register, forgot username, and forgot password. This means that malicious users can't create bots that spam our database with fake accounts

and also can't use our email service to spam other people's emails under our name. Moving on to what I implemented for the database, I created the table used to contain spots that are currently being swapped, and also the table called spots history that contains all the spots that have been deleted by the spot trading implemented by David. Beyond that, I implemented several cascading triggers in the database that handle all the various bookkeeping. As soon as a spot is deleted, depending on the conditions it was deleted in all the token handling, rating updates, and history keeping is done using this set of triggers. I did this in order to remove the dependency on users clients issuing SQL queries to the database to make it less likely for database errors to occur. Essentially, all we rely on is the users hitting complete slash cancel and then the Delete query going through from their client. This is opposed to issuing every update query through the client using PHP, which would mean that if the client lost connection halfway through completing a trade, a small piece of the completion might not occur, such as token gain or loss or rating updating overall, this just prevents spa trading from being manipulated by users to keep tokens or otherwise tamper with the trade by intentionally disconnecting. There are some images here on the slide. On the left side, we have the captcha. This is for our account creation. It's essentially just reCAPTCHA. Using Google API. On the right we have a little diagram showing how a user completes a spot and how the different triggers initiate. So essentially, as soon as a user completes or cancels the spot, the update token triggers. This distributes tokens to the people who offered a spot and takes away tokens from the people who requested them, but only if the spot was completed correctly.

Tony:
Moving on to the features that I completed. First thing first was user registration. Of course, this would allow any user to create and register an account with our application, as well as make their user profile. Right after that I started working on profile update. This would allow users to update their profile with maybe requesting a new username change, changing their first and last name if they got married or something as such, as well as update their vehicle's information if they happen to buy a new car or just started using a different vehicle. Thanks to the help of David, we were able to also implement uploading a new picture and having that refresh and automatically, in real time show the new picture upon submission. The next thing was administrator database table views essentially without having to sign into our database through the service side every single time, you can view everything that is contained within our database within the application itself. So long as you weren't a normal user. You would have to be either an administrator or just a normal employee. For forgot username, a user can sign into our web application either using their email address or their username, but in the event that they don't want to use their email address and insist on trying to find their username, they can always request their username be sent to them. Just as well with our password reset page, when a user enters their email address into the text field, if that matches an email within our database, it's going to send that specific email a unique token that gets generated. With that token, the user will have five minutes to be able to reset their

password. Lastly would be our geolocation map when a user gets paired with the parking spot, a map will show them where their lot is, and it will begin zooming in the closer you get to that spot, which is going to help the user find the parking spot that was assigned to them.

Abraham:
So for the design of the app, I started off by first making some rough sketches on paper. And then after that, I used Adobe XD to create a few different prototypes of how the app would look. And during this time, I also designed the logo for the app. And I tried a ton of different versions before finally setting on the one we have. But after this, I used CSS. And I used bootstrap to create the grid layout for the app. And I decided to go with a mobile first approach, since this is primarily meant to be a mobile app. And because it would also make development easier since we could just use the desktop version for any like testing or whatever we have to do, and not really have to worry too much about making sure it looks good on mobile. For the design of the app, I decided to go with a dark theme, because it's easier on the eyes. For the other elements like the buttons and the order of layouts, my main priority with the design was to make sure that it was as straightforward as possible and easy to use. And I made sure that there was a clear hierarchy of elements, like for example, in the card that shows up when you're paired with someone, there's a clear separation between like the account info and the two different buttons. And I wanted to do that. So there wouldn't be too much confusion about how to use the app as because I just wanted to make sure that it was as clean and intuitive as possible. That way it's easier for people to use and more likely for people to use, and just minimize frustration with it as much as possible. So yeah, that's what I did for the design of the app.

David:
So now I'm going to go over a few challenges that we faced while implementing these features. First one, I was working on the photo upload feature, I started off with a PHP script. I was trying to write my own script to upload these photos into a folder in our directory on Odin. But ultimately, that didn't work because of some permission errors. And I ended up using the Imgur API instead. That ended up being pretty easy to implement and a lot quicker to retrieve and upload folders. Next, I had a couple issues with the spot queue algorithm at first, just learning how to use Ajax to send and retrieve data from the server. That was a challenge in itself. And also just getting both requester and offeror spots working together was quite a big challenge.

Tony:
The next set of challenges we face was with geolocation mapping. Essentially, the issue was how do we transfer our database content into our JavaScript that way Google Maps API can automatically adjust properly. Once we figured that out, the next set of issues were how to show the markers appearing for both the requests that you were matched with as well as with your current position. When we got it all working, it seemed to be

perfectly fine until you started driving towards the location, it would add up markers. So essentially, it would start leaving a trail of markers behind you as you moved. This was then fixed with the help of David by making a global variable that would just clear it upon setting your current position again. The other one is password reset. This was challenging because most of the guides or just any article I found online had at minimum three pages. So luckily, eventually, I was able to condense this into a single page to generate the user's unique token, as well as use Odin server email server to be able to send the user an email with that link with the token in a get request. And in terms of security, the token expires within five minutes so that someone doesn't try to access the token later on or try to spoof it and essentially change a user's password. The last one would be SQL triggers. Terry was having issues implementing this or getting it to work perfectly, just because of how many different times we access our database when trying to match a user with their spot as well as removing the tokens. But eventually, Terry was able to figure out how to get properly set up on their triggers. That way, when a user's spot request gets deleted, it removes or adds tokens appropriately to the user's account.

David:
As we were implementing some features we noticed we had to make a few changes, the first of which was scrapping an old timer idea that we had this idea essentially didn't work because they required the poster to post their spot before they were even at their vehicle. And this ultimately would have caused issues with the geolocation feature that we implemented. The next feature that we had to change was the geolocation map itself. We made it so that the users would need to request a spot when they were within a certain radius from campus, just so we can ensure that these transactions are quicker. And this also caused us to no longer need a GPS navigation for the requester. And lastly, we wanted to add a rating system, which required the use of triggers in our SQL database, just so users can know who is reliable.

Abraham:
Now, to recap, how is this going to help people? Well, it's going to help people by saving them a lot of stress, like not having to worry about, oh, how early do I need to leave for school because I may not be able to find parking, and it's gonna save them a lot of time. So they don't have to drive all over campus in circles, just trying to find a spot somewhere. And I think it just, it's gonna help a lot of people, especially during like the busiest hours at school. So what did we learn with this app, we learned a lot of web design like PHP, HTML, CSS, we learned algorithm analysis, and we learned how to implement databases into PHP to use on the web and to store and collect data.

Tony:
And now for our future plans, the first one being adding timers for both the requester and the poster. This way, let's say that someone posts their parking spot during non peak times, and no one accepts their spot or no one gets paired up. In this case, the

poster is out of luck, because they're not going to be able to get their token back. So in this case, we would add a timer for about five minutes. That way, if five minutes passed, the poster no longer has to remain at their parking spot and can leave with no worries, of not getting a token. This way, the spot gets deleted, and their token is allocated. As well as, this will protect the requesters so that after those five minutes, as soon as the poster has left, that parking spot is no longer available, so someone doesn't end up getting paired with it much later in the day. And now that spots no longer guaranteed As well for the requester, so in this case, the requester would have about five minutes to reach their destination to their parking spot. That way that gives ample time for them to arrive. And that way the poster isn't waiting too long for the person to arrive to their spot. As well as if that timer was to expire again, the requester would then be notified that that parking spot is no longer guaranteed. So they can choose to find a new one or just cancel the request or the matching in general. Our additional future plan would be expandability. At the moment, our application is geared more specifically toward CSUB more as a test case. So in the future, we can end up making this application be adaptable to every single type of CSU or UC system, where parking might be issues as well as potential stadiums or other type of concerts and events where parking might not be guaranteed or as easily available for users. So this is where we'll have to implement and how to be able to choose locations based on where you're at or where you're planning to park as well, in general, just expanding our app to multiple different venues.

And now on to our demo, starting with Abraham showing us the registration page.

Abraham:
Then we are going to put in our information. So I'm gonna put in my name, last name, create a username, enter your email, aldana, just gunna put this. Password - I'm just going to put p a s s. Then you put in your car information. So I have a Honda Civic, 2014. I'm just going to put in some random numbers here. And color, I have a red car, then you just click I am not a robot here to complete the captcha and then create account. Now to upload an image of the car, you're just gonna click Choose File and click your image. And there we go, upload. And it's up. Now to login. You just put in your email or username, and then the password and login.

Terry:
Hey, everybody, this is Terry. I'm just going to be demoing our forgot username and for that password for you. When you're navigating to the Forgot username section, you're asked for an email and then also presented with a recapture. If you simply put it in email and don't complete the recapture when you hit submit, it won't actually send any kind of email. You won't receive any kind of notification or anything like that. That's to prevent malicious users from using our email services for, well bad things, obviously, I'm just gonna go ahead and complete the reCAPTCHA. We're gonna see if we're able to get an email, you're prompted with a little notification, just letting you know that an email has been sent to your email address. We should see something pop up anytime now. There

it is. So, on the email side, you'll just get a little message chain. Hey, if you forgot your username, this is your username. Now that you have it, you're able to log in. On our website, you're also able to log in with just your email. But this is just in case people want their username to be able to log in later on.

Now I'm going to move on to the Forgot Password section, it works much the same, it's essentially the same concept. Once again, if you don't fill out the reCAPTCHA, it's not going to send you any kind of email, or any kind of reset link or anything like that. But if you do fill out the reCAPTCHA, it's gonna give you another prompt just letting you know, hey, we're gonna send you an email for that. Button refresh, this is the email that you get for the password reset. It also prompts you that if you didn't request for this password reset, you should go and change your password on your account. Because that means somebody is trying to get into your account, obviously, to be able to reset it, you just go and hit reset, this is going to give you the new reset. So we'll just reset it to something like one of my used passwords. Password has been successfully changed, go ahead and try to log in. So we'll try to just go ahead and log in really quickly. And just like that are able to access the website once again.

David:

Okay, so now we're going to demo the parking spot trade feature that we have implemented. So when I click find a spot as the requester, it will take me to another page that'll enter me into a parking spot request queue. And it'll be continuously searching for available spots in the database. And once a spot becomes available, it'll pair me. And I believe Tony is on the other end, about to offer up a spot.

Tony:
So let's say in this case, I'm about to leave campus and I no longer need my parking spot so I can give it up so I can get a token back. If that is I ended up using it that day.

So in this case, I'll select the button that says give a spot and as you can see, it says that I am not yet offering a parking spot. So I just have to choose whichever lot we belong or I am currently parking so let's choose lot F. Then I'm going to select offer my spot and as you can see, it says my spot has been posted.

David:
So now, both of our pages should have updated as they're both continuously checking for your parking spot matches. I'm shown with a few details of Tony's car, such as the color, model, make, and year, as well as the license plate digits at the end. So when I click on the View Details button, it will take me to another page where it'll allow me to share my location. It'll give me Tony's details, as well. And if I scroll down, it'll also show me their parking spot location. Generally, this is more useful once we're actually near the parking lot that we're trying to reach. But for right now, it zooms in once we get closer to the destination.

Tony:
And then on my site, let's say that for some reason, I end up going to the previous page or the page just closes out, times out or anything like that. If I was to go ahead and select the give a spot again, it would already show that I'm currently matched - which in this case, it would take me back to the page where it shows the person that I was matched with. And it's essentially at this point waiting for the requester to confirm whether or not they got the spot.

David:
And so once I go ahead and click Complete the trade, it'll basically exchange the tokens between me and Tony. I will lose a token since I've just requested a spot and Tony should gain a token onto his account. As you can see here, I started off with two tokens when I logged in, and now I currently have one.

Tony:
And on my screen you can see that the trade has been completed. So if I go back to the homepage, I started off with 10. And now I have 11. And with all the demos of our application that concludes our video. Thanks for watching.