1. Install NodeJS in your machine and copy folder to any local drive.
2. Open folder in a terminal, VSCode or otherwise, and type “npm install” to install all dependencies listed in package.json file and to install devDependencies, type “npm install <package\_name> --save-dev”.
3. **To connect to MongoDB:**
   1. Log in to mongodb.com and create a cluster inside a project.
   2. Go to ‘Database Access’ in the MongoDb console and add a user. Remember the username and the password as it will be used later.
   3. Then go to ‘collections’ and create a new collection with name “test” and a database with name “users”.
   4. Then click on ‘connect’ followed by ‘connect your application option’.
   5. Go to dev.js file in config folder of the project and put the username and password created in step2 in place of “carrental:carrental” in the MongoDB object. This should connect the database to the project.
   6. If not, after all the 5 steps above go to network access and specify 127.0.0.0/0 in the IP Access List.
4. **To connect to Facebook Authentication:**
   1. Go to developers.facebook.com and log in/ sign up.
   2. Then go to ‘My App’ option and click on “Create App”.
   3. Choose ‘Facebook Login’ from the dashboard that comes up and click on ‘Set Up’.
   4. Go to settings and then look for App ID and App secret keys.
   5. Go to dev.js file and put those keys in place of “FBAppID” and “FBAppSECRET”.
5. **To connect to AWS Storage:**
   1. Go to AWS Console and create a free tier account. You needn’t need to enable billing for this project.
   2. Then go to S3 and create a bucket.
   3. Then go to IAM console and select Policies tab. In the policies tab, click on ‘create new policy’ and choose S3 in the “Service” option after that. Also, if prompted, select ‘All manual S3 actions’ in the actions menu. Then in the Resources section, add ARN in both bucket and object parts, where you’ll need to specify the name of the bucket you created in the previous step. This should create the Policy required for this project.
   4. Then go to User section in IAM and create a new user with “Programmatic Access”, where you’ll get an option to attach the existing policy created in the previous step with this user.
   5. After creating the user, you’ll be presented with Access Key Id and Secret Access Key.
   6. Copy these keys to ‘AWSAccessID’ and ‘AWSSecretID’ objects resp. in dev.js file.
6. **To connect to Google Maps:**
   1. Go to Google Cloud Platform and sign in/ sign up.
   2. Go to dashboard and click on enable APIs. Then enable Maps Javascript API, Places API, Geolocation API and Geocode API.
   3. Then in dashboard, go to Credentials and click on ‘Create API key’. Follow the steps shows to create an API key for google services.
   4. Copy that key into “GeocodeAPIKey” and “GeolocationAPIKey” objects in the dev.js file to enable Google Map Services.
   5. You may be asked to create a billing account for the APIs to run smoothly, so you can create a billing account. Don’t worry as nothing will be charged from that account as it is just a process to verify that you’re not a robot. You can later disable or remove that billing account too.
7. **To connect to Stripe:**
   1. Go to stripe.com and login/ sign up.
   2. Then in the dashboard, click on “Developers” option and then go to API Keys path.
   3. Copy the publishable and secret keys into “StripePublishableKey” and “StripeSecretKey” objects in the dev.js file.
8. To run the project, type “nodemon” or “npm run start”, and then go to localhost:3000 in your browser.
9. Also you have to copy Google Map API key into the link in ‘googlemap.handlebars’ file and ‘listcar2.handlebars’ file.
10. dev.js file is present in config folder of the project directory and all the ‘.handlebars’ files are present in the views directory.
11. You can style the pages locally by specifying style tags in all of the handlebars files or you can apply styles globally to all the pages by specifying style tags in “styles.css” present in Public directory.