npm create vite@latest

npm install react-data-table-component styled-components axios react-icons react-router-dom tailwindcss postcss autoprefixer

create folder inside that folder

npm init -y

npm install bcrypt cors express jsonwebtoken mongoose multer nodemon path

create index.js file

add to package.json

type module

start nodemon --env-file=.env index.js

create .env file

1. We will make pages for our project pages to route and components for reusability and utils for our logic and connection code
2. To use the .env file import dotenv and config() that
3. Import cors means cross origin resourse sharing and it act as middleware which allow user to make request from any port or address
4. app.use(express.json()); it will parse and convert the data in node js in Jason format
5. then we will create the model folder in which we make the model for the User
6. in role fiel we use enum property because **enum: ['admin', 'employee']**: This limits the role field to only accept the values 'admin' or 'employee'. Any other value would result in a validation error when saving the document
7.  **mongoose.model('User', userSchema)**:
8. **mongoose.model**: This is a method provided by Mongoose, which is a popular library for MongoDB and Node.js. It creates a model based on the schema you've defined.
9. **'User'**: The first argument is the name of the model. Here, it's 'User', which will represent a collection in your MongoDB database. Mongoose will automatically pluralize this name to create the collection name ('users' in this case).
10. **userSchema**: The second argument is the schema that defines the structure of the documents within the 'User' collection. It contains information such as the fields, data types, and validation rules for each user.
11.  **const User =**:
12. This line assigns the created model to the User constant. Now, you can use the User variable to interact with the User collection in MongoDB (e.g., creating, reading, updating, and deleting users).
13. We will try to create the instance in our database by making userseed.js file in which we connect it to db and save the user and check if our code works or nt
14. Now we will come to the frontend side to create routes and browserRoutes
15. 
16. Now creating the login controller and the login UI page for the login
17. Now we are using the create context and if the login is successful and the user is admin it navigate to the admin dashboard else it navigate to the employee dashboard
18. We use private routes for to check if the user is already logged in or not
19. The end property in the navlink ensures that is the {to} path is exactly same to the url then it isactive if we not apply the end property then the partial url also mathches give the result. Basic it ensures the url and the (to) path should be exactly same
20. And learn and study more about the outlet which give the property to render a small portion of components to the previously rendered ppage
21. What is Outlet in React Router?
22. In React Router, the <Outlet> component acts as a placeholder in a parent route, rendering the child route's components when a nested route is matched.
23. It helps implement nested routing, allowing you to display components for both the parent route and its child routes in the same layout.