**A brief write-up on their approach and any assumptions made**

FastAPI's great performance, ease of use, and integrated support for asynchronous programming led to its selection. Furthermore, and this is useful for API projects, it offers automatic OpenAPI documentation production.  
Verification  
  
implemented JWT tokens for user authentication and registration. Bcrypt is used to securely hash passwords, guaranteeing that user credentials are kept private.  
Management of Databases:  
  
Because of its strength and dependability, PostgreSQL was chosen as the relational database.  
Database operations and migrations were managed more easily since SQLAlchemy ORM was utilized to communicate with the database.  
Management of Tasks:  
  
To ensure that users may add, read, update, and delete tasks, CRUD procedures were introduced for task management.  
Users are given tasks, and authentication is used to ensure ownership.

Filtering was put into place according to task status, priority, and due date.  
Users can now search tasks by title or description thanks to the addition of search capability.  
Dockerization:  
  
Docker-compose and a Dockerfile.YML was made available to streamline the application's setup and deployment process while guaranteeing uniformity in various settings.  
Record-keeping:  
  
A Postman collection was used to generate API documentation that described the accessible endpoints and the anticipated inputs and results.

In order to create basic user authentication, it was assumed that each user had the same level of access. Although it was not included in the main project, role-based access control (Admin vs. User, for example) was seen to be a possible improvement.  
Ownership of the Task:  
  
It is assumed that each job belongs to a single user, and that only the owner is able to carry out CRUD activities on their assignments.  
Configuring the Environment:  
  
Users are expected to have Docker installed on their PCs in order to execute the application with the least amount of configuration and set up the database.

A somewhat small-scale application is assumed by the current implementation. Although the architecture allows for simple searching and filtering, it was not intended to handle massive dataset processing or more intricate query optimization.  
Safety:  
  
Assuming HTTPS will be used in production to protect token transmission, JWT tokens are utilized for authentication. Furthermore, it is expected that the production environment securely maintains the secret key required for JWT.  
An overview of the design choices and factors taken into account during the task management API's development, as well as the presumptions that influenced its implementation, are given in this article.