BLG 361E Term Project – ITUnder

Sahin Olut
Hamza Ali Tas

Department of Computer Engineering
Istanbul Technical University

December 14, 2017

Outline

1. About the ITUnder
   - Aim of project
   - Capabilities of ITUnder
2. Technical details
   - Development Environment
   - E/R Diagrams
   - Our technology stack
   - Some code from project

Outline

1. About the ITUnder
   - Aim of project
   - Capabilities of ITUnder
2. Technical details
   - Development Environment
   - E/R Diagrams
   - Our technology stack
   - Some code from project

Aim of project

- We thought ITU students have hard time during studying exams.

Aim of project

- We thought ITU students have hard time during studying exams.
- By ITUnder, ITU students can synchronize their Ninova and form study group for their academic work.

Aim of project

- We thought ITU students have hard time during studying exams.
- By ITUnder, ITU students can synchronize their Ninova and form study group for their academic work.
- ITU account can be used safely and it makes easier it to use.

Outline

1. About the ITUnder
   - Aim of project
   - Capabilities of ITUnder
2. Technical details
   - Development Environment
   - E/R Diagrams
   - Our technology stack
   - Some code from project

Functionalities

- Users can register, remove, update their credentials.

Functionalities

- Users can register, remove, update their credentials.
- Users can form study groups, chat groups, add homework to their schedule etc...

Functionalities

- Users can register, remove, update their credentials.
- Users can form study groups, chat groups, add homework to their schedule etc...
- Students can anonymously join study group or make comments on others profile.

Some demo...

- Some of capabilities are demonstrated.

Outline

1. About the ITUnder
   - Aim of project
   - Capabilities of ITUnder
2. Technical details
   - Development Environment
   - E/R Diagrams
   - Our technology stack
   - Some code from project

Dev. environment

- We used Docker instead of Vagrant because Vagrant consumes more system resources than Docker and also we can deploy the docker instance directly if we want to turn our project to product.

Dev. environment

- We used Docker instead of Vagrant because Vagrant consumes more system resources than Docker and also we can deploy the docker instance directly if we want to turn our project to product.
- For consistency and improve the readability of code, we sticked to pep8 formatting and camel case convention is kept throughout the project.

Dev. environment

- We used Docker instead of Vagrant because Vagrant consumes more system resources than Docker and also we can deploy the docker instance directly if we want to turn our project to product.
- For consistency and improve the readability of code, we sticked to pep8 formatting and camel case convention is kept throughout the project.
- Both of us were already using same text editor(VSCode) and Unix based operating systems, therefore building the dev environment in our system did not cause an issue.

Outline

1. About the ITUnder
   - Aim of project
   - Capabilities of ITUnder
2. Technical details
   - Development Environment
   - E/R Diagrams
   - Our technology stack
   - Some code from project

E/R Diagrams

- content... ER diagramlari koy buraya

Outline

1. About the ITUnder
   - Aim of project
   - Capabilities of ITUnder
2. Technical details
   - Development Environment
   - E/R Diagrams
   - Our technology stack
   - Some code from project

Back-end systems

- We created a RESTful api for back-end services so we can extend our Web App to mobile or desktop easily. For instant messaging(chatgroup) and storing messages, we used Redis database and Node.js socket.io framework since we had already many entities in PostgreSQL.

Front end

- In front end, Hamza recommended to use React.js framework and we coded the front end of web app in React framework.
- ESlint and Babel are used for styling and backward compatibility.

Outline

1. About the ITUnder
   - Aim of project
   - Capabilities of ITUnder
2. Technical details
   - Development Environment
   - E/R Diagrams
   - Our technology stack
   - Some code from project

Database connection decorator

```
db = SQLAlchemy(app)
def db_factory_func(fn):
    """
    DB connection decorator.
    """
    @wraps(fn)
    def wrapper(*args, **kw):
        try:
            conn = db.engine.connect()
            result = fn(conn=conn, *args, **kw)
            if result is not None:
                return [dict(r) for r in result]
            return result
        finally:
            if conn is not None:
                conn.close()
    return wrapper
```

Base class for every model

```
class BaseModel:
"""
Base class for the data manipulation and database operations.
"""
    def __init__(self, table, fields, primary_key=None, init_table=False):

    @db_factory_func
    def __init_table(self, conn):
    #  Initializes the database.

    @db_factory_func
    def create(self, conn, data):
    #  Insert operation

    @db_factory_func
    def find(self, conn, query="", limit=0, sort_by="", return_cols=None):
    #  Read operation

    def find_one(self, query=""):
        return self.find(query=query, limit=1)

    def find_by_id(self, _id):
        return self.find_one(query="id=%s" % _id)

    @db_factory_func
    def update(self, conn, data={}, query="", returning_id=True):
    #  Update operation

    @db_factory_func
    def delete(self, conn, query="", returning_id=True):
```

Summary

- We learned the lifecycles of a project and enjoyed. In addition, we learned new technologies like Docker, node.js, React.js, Redis, PosgreSQL...
- We understood how to use relational databases efficiently and how to write efficient SQL queries.
- Thank you for listening.