Author

Name : S Manish
Roll No: 21f3002911
Student Email: 21f3002911@student.onlinedegree.iitm.ac.in
About: I am a creative, ambitious and determined person.

Description

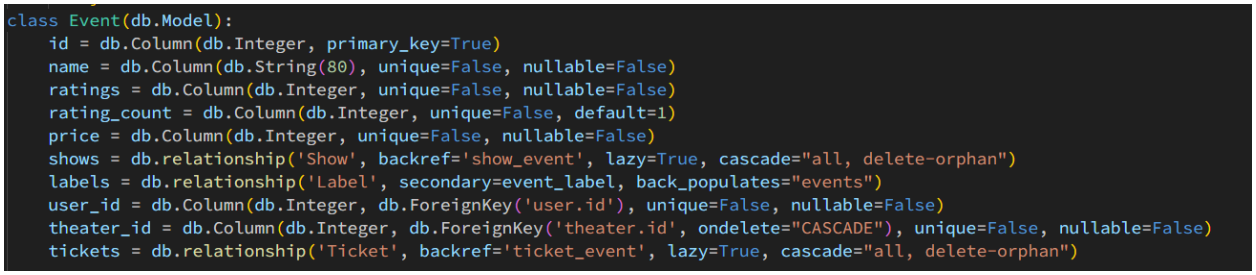
We are expected to create an app that can be used as a platform to book tickets. Admins of the platforms have the permission to create and manage Venues and Events. Users of the platform have the ability to book tickets and view them

Technologies Used

Flask,
VueJS (For UI)
Flask-Restful (For api services)
Flask-SQLAlchemy (For databases),
Flask-Mail (For Mailing)
Mailhog(For SMTP Testing)
PyJWT (For login system using JSON Web Tokens),
Celery (For Cronjobs),
Redis (For message broker and caching)
Jinja2 (For Email Templating)
Flask-Caching (For performance and caching purposes)
Pandas (For CSV export)

DB Schema Design

```python
class Theater(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), unique=False, nullable=False)
    place = db.Column(db.String, unique=False, nullable=False)
    city = db.Column(db.String, db.ForeignKey('city.name'), unique=False, nullable=False)
    capacity = db.Column(db.Integer, unique=False, nullable=False)
    shows = db.relationship('Show', backref='show_theater', lazy="dynamic", cascade="all, delete-orphan")
    events = db.relationship('Event', backref='event_theater', lazy=True, cascade="all, delete-orphan")
    tickets = db.relationship('Ticket', backref='ticket_theater', lazy=True, cascade="all, delete-orphan")
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), unique=False, nullable=False)
```

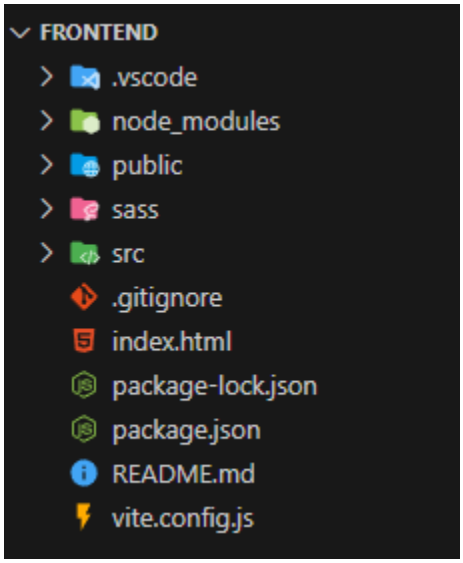


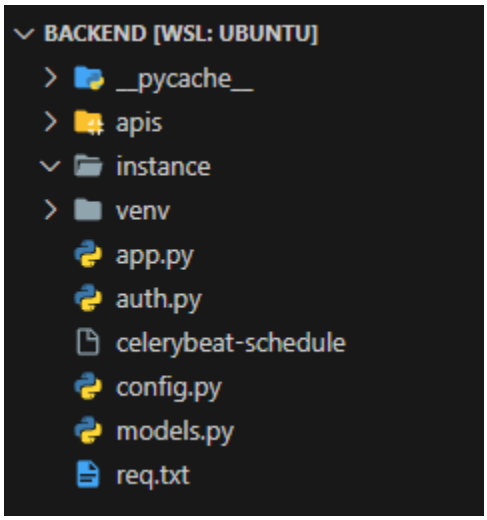
```python
class Event(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), unique=False, nullable=False)
    ratings = db.Column(db.Integer, unique=False, nullable=False)
    rating_count = db.Column(db.Integer, unique=False, default=1)
    price = db.Column(db.Integer, unique=False, nullable=False)
    shows = db.relationship('Show', backref='show_event', lazy=True, cascade="all, delete-orphan")
    labels = db.relationship('Label', secondary=event_label, back_populates="events")
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), unique=False, nullable=False)
    theater_id = db.Column(db.Integer, db.ForeignKey('theater.id', ondelete="CASCADE"), unique=False, nullable=False)
    tickets = db.relationship('Ticket', backref='ticket_event', lazy=True, cascade="all, delete-orphan")
```

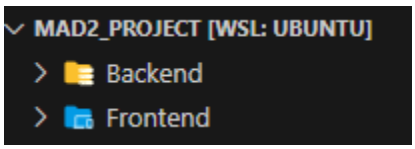
```python
class Show(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    event_id = db.Column(db.ForeignKey('event.id', ondelete="CASCADE"), unique=False, nullable=False)
    theater_id = db.Column(db.ForeignKey('theater.id', ondelete="CASCADE"), unique=False, nullable=False)
    start_time = db.Column(db.DateTime, unique=False, nullable=False)
    date = db.Column(db.DateTime, unique=False, nullable=False)
    end_time = db.Column(db.DateTime, unique=False, nullable=False)
    available_seats = db.Column(db.Integer, unique=False, nullable=False)
    tickets = db.relationship('Ticket', backref='show_ticket', lazy=True, cascade="all, delete-orphan")
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), unique=False, nullable=False)
```

Architecture and Features

Structure:







Features Implemented:
→Creating and managing theaters and shows
→ Booking tickets
→ Cronjobs for users and admins
→ Caching with redis for performance

Video

[Video Link to MAD2 Video Report]