# Bicycle Auction WebApp

CS 699 Project Report

**Abhishek Kumar Gupta 22M0826**

**Osim Abes 22M0825**


Department of Computer Science and Engineering

IIT Bombay

Guide: Prof.Bhaskaran Raman

# Contents

# 1 Description

Bicycles are the primary mode of transportation inside the IIT Bombay campus and remain so post-lockdown. The other alternative available is auto-rickshaws because personalized motor vehicles aren't allowed. But Rickshaws aren't environmentally friendly in the first place and also cause traffic issues such as congestion and even accidents in the worst case. This increases the need for bicycles even more, but students, especially the freshers, find it difficult to identify proper channels for purchasing cycles because either they are too expensive or are not reliable enough.

Hostels, too, conduct auctions of bicycles, but the problem with these auctions is that students don't get to know about these auctions, and on top of that, the process itself isn't transparent enough, and exorbitant prices are charged. This creates the need for a centralised, transparent and reliable system for conducting these auctions.

We have tried to provide an amicable solution to this issue by building a web application for conducting e-auctions of bicycles. Any student/hostel of the institute can create its own bidding slot and upload details like pictures, specifications and the expected price of the bicycle. A buyer can bid prices above a seller-defined base price within the bidding period set by the seller. At the end of the period, the highest bidder gets the bicycle.

## 2 What we have achieved

We have been able to build a Responsive and Dynamic platform implementing REST Architecture using Django Framework at the Back end , React JS to build the User Interface and PostgreSQL for Database which is ready to be deployed on public servers.

Our platform allows users to create their personalised accounts, and the account created is authenticated by OAuth to ascertain the proper identity of the user and help protect against malicious users. After account creation, Users can enter the marketplace and will be able to take part in bidding activities. Users can put up their bicycles for auction by creating a post, including a description of the entity, pictures and the expected amount for a buyout or range of bidding prices. Also, the user would be able to put up offers on the auction table for buying any bicycle hosted in the marketplace.

Each user is provided with a dashboard where all information about their bidding activities is displayed, like offered bids, winning bids and user posts. After a user wins an auction, they are provided with seller information and a unique one-time verification code, which can be used to redeem the buying privileges acquired in the auction.

We have also provided an excellent and compact visual interface for users to keep real-time track of current trends in the bidding process of their favourite bicycles. Also, the most trending bikes on the site are displayed to the user on the platform's landing page.

# 3 Scope for Future Work

- Advanced Search Functionality
- LDAP integration to make the platform IIT Bombay specific
- Opening up a payment gateway to make the platform self-sufficient for an entire transaction
- Integration of AI into the platform for detecting spam users

# 4 Documentation

This project is divided into two parts, Frontend and Backend which uses REST Architecture for facilitating interaction between them.

Techstack used:

- ReactJS, Material UI for Frontend
- Auth0 for Authentication
- Apache Echarts for Plots
- Django REST framework for Backend
- Postgres for Database

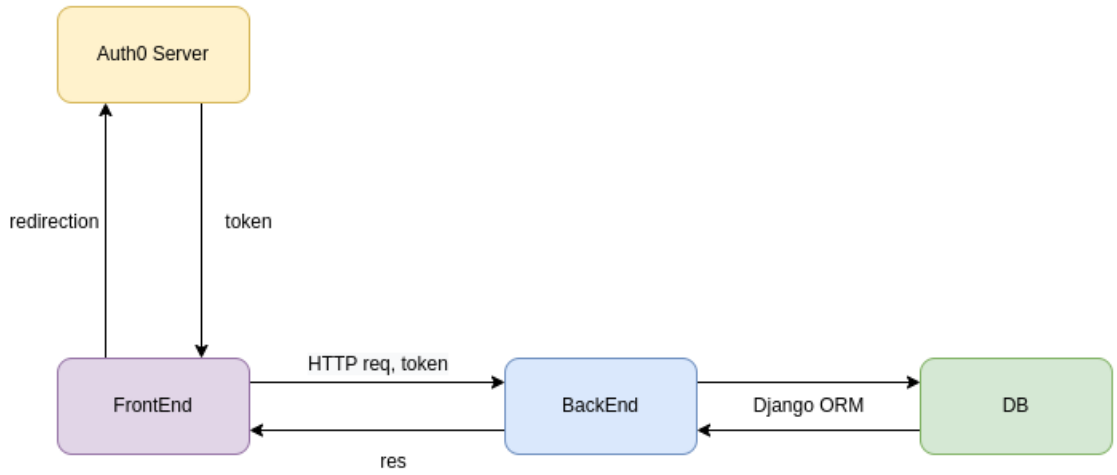


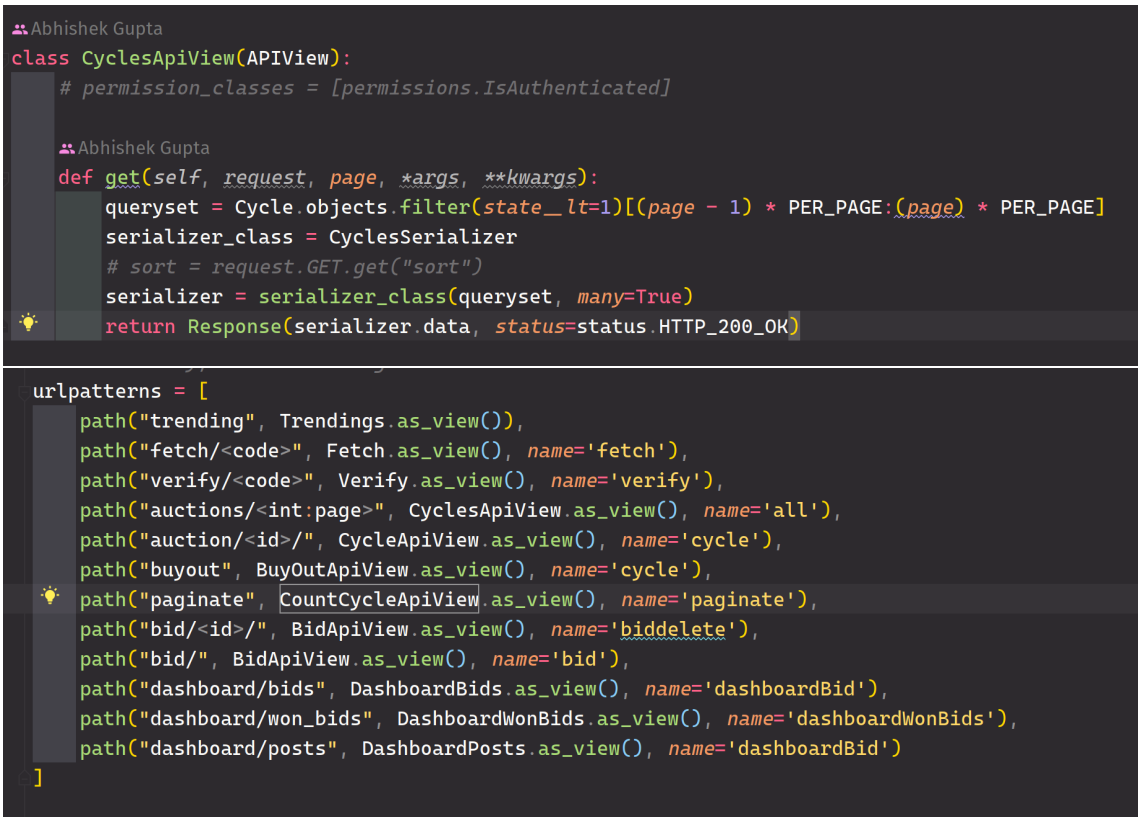

Figure 1: Flowchart

## 4.1 Backend

Backend code can be found in server directory which follows standard django pattern, some of which are described below:

- manage.py is the main file via which many functionalities are executed like launching the backend.
- docker-compose.yml has Postgres database server configuration
- The backend is further divided into two sub apps jobs and restapi.
- The jobs sub app contains two scheduled tasks which run on regular intervals and the functions can be found in task.py. The trending task updates the trending cycles section in the database which executes every hour. The other task is to update_bidding_state which runs every minute and disables the auction state for those cycles which have passed their respective deadlines or have been purchased.

- The restapi app contains all the endpoints used by the frontend app. It mainly consists of two segments. First, the Class base view is defined inside view.py file which contains all logic on how to handle various requests. Second, these views are mapped to endpoint urls in the the url.py file. The Schema of the Database can be found inside models.py .



```
Abhishek Gupta
class CyclesApiView(APIView):
    # permission_classes = [permissions.IsAuthenticated]

    Abhishek Gupta
    def get(self, request, page, *args, **kwargs):
        queryset = Cycle.objects.filter(state__lt=1)[(page - 1) * PER_PAGE:(page) * PER_PAGE]
        serializer_class = CyclesSerializer
        # sort = request.GET.get("sort")
        serializer = serializer_class(queryset, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)
```

```
urlpatterns = [
    path("trending", Trendings.as_view()),
    path("fetch/<code>", Fetch.as_view(), name='fetch'),
    path("verify/<code>", Verify.as_view(), name='verify'),
    path("auctions/<int:page>", CyclesApiView.as_view(), name='all'),
    path("auction/<id>/", CycleApiView.as_view(), name='cycle'),
    path("buyout", BuyOutApiView.as_view(), name='cycle'),
    path("paginate", CountCycleApiView.as_view(), name='paginate'),
    path("bid/<id>/", BidApiView.as_view(), name='biddelete'),
    path("bid/", BidApiView.as_view(), name='bid'),
    path("dashboard/bids", DashboardBids.as_view(), name='dashboardBid'),
    path("dashboard/won_bids", DashboardWonBids.as_view(), name='dashboardWonBids'),
    path("dashboard/posts", DashboardPosts.as_view(), name='dashboardBid')
]
```

Figure 2: View and URL mapping

## 4.2 Frontend

Frontend code can be found inside the the client directory.

- App.js is the main file in which all modules are imported.
- src/components directory contains all the UI components like buttons, modals, inputs, loader, etc., that are used across various pages.
- The backend apis are called inside these components for handling various HTTP requests.
- Authentication of user identity is implemented using Auth0. Before participating in an auction user must register into the platform.
- A plot of the bidding history of a specific auction is displayed and is implemented using Apache Echarts.

## 4.3 Database Schema and ER Diagram

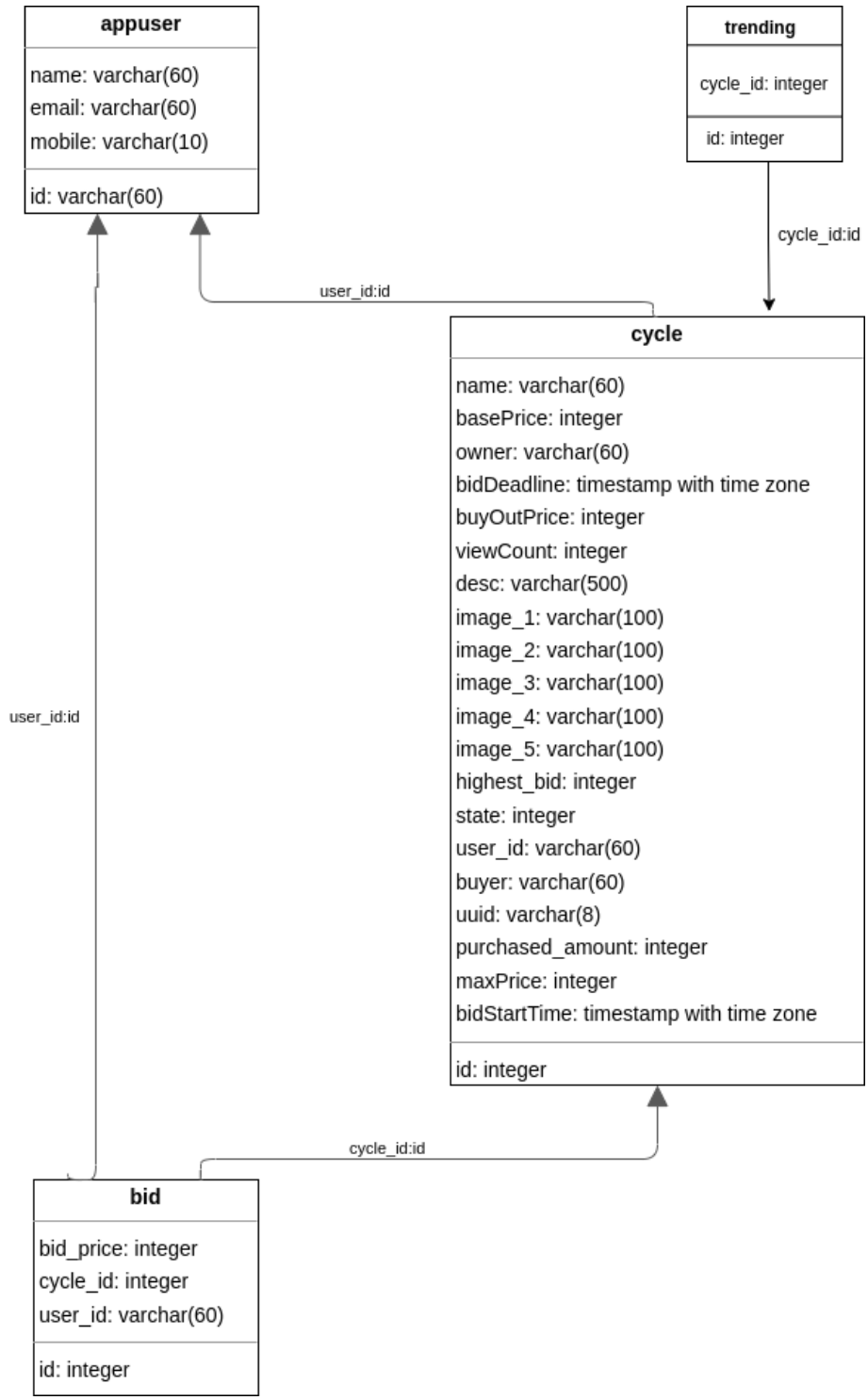



Figure 3: ER Diagram

# 5 Running instructions Backend Server

- docker compose up -d
- pip install -r requirements.txt
- python manage.py makemigrations
- python manage.py migrate
- sudo systemctl start rabbitmq-server
- sudo systemctl enable rabbitmq-server
- run celery -A server worker -l debug on one tab
- run celery -A server beat -l INFO on another tab
- python manage.py runserver

# 6 Running instructions Frontend

- npm install
- npm start
- Now open localhost:3000 and you are good to go.