Project K5 Progress

Catalyzing the Galvanization of Filipinos in the Contribution to Targeting the Paris Agreement Alignment through the Implementation of a Web-based Application

Aidre Love S. Cabrera
Project Lead | BS Computer Science

RATIONALE

The motivating need for Project K5 Progress arises from the dire climate crisis faced not by the Philippines only [1], but also globally [2]. The country has been grappling with increasingly frequent and severe natural disasters, including typhoons, flooding, and droughts [3], [4]. Rising sea levels threaten coastal communities [5], while deforestation and habitat destruction contribute to biodiversity loss [3]. These pressing issues have brought the urgency of climate change to the forefront of Philippine national concerns [6], [7]. Additionally, the Philippines, as a signatory to The Paris Agreement, has committed to taking action to mitigate climate change [8]. However, there is a significant gap between promises and commitment and effective cooperation, awareness, and movement in climate action context.

It is certain that the Philippines has witnessed the devastation caused by climate change firsthand. The nation has been named one of the most vulnerable countries to climate change due to its geographical location and socioeconomic factors [6]. International reports and agreements, such as The Paris Agreement and the Intergovernmental Panel on Climate Change (IPCC) assessments [8], have emphasized the urgency of addressing climate change on a global scale. Hence, it is unequivocal that there is a need for Filipinos to be more aware of the climate crisis and to take action to reduce greenhouse gas emissions. However, many people are not sure what they can do or where to start. The need for practical, data-driven climate solutions that engage communities is apparent. And this is where the proposed project, Project K5 Progress, seeks to respond to this context by offering a tangible solution to accelerate climate action.

Key technical issues underlying the motivating problem include the lack of tailored climate change contextualization, limited public awareness, and engagement in climate issues, and the absence of a local user-friendly tailored for platform for reporting and monitoring progress. Climate data is often fragmented and indirect, making it challenging for decision-makers, the citizens themselves, and the public to access timely and reliable information. Public awareness and engagement are crucial for building a nationwide consensus on climate action.

To bridge these gaps, a project is necessary to be proposed. Hence, that project is called K5 Progress, which aims to create a platform that empowers individuals and communities to actively participate in climate action while providing them with the necessary data and tools. Project K5 Progress is a web application that aims to catalyze climate action in the Philippines by providing a local platform for Filipinos to monitor, report, and promote progress in the country's efforts to align with The Paris Agreement on climate change. With this, the project lead argues that it might be one of the contributors in galvanizing the youth, if not, every Filipinos in the climate crisis.

EMPATHIZE

In the context of the Project K5, the project came to be through the empathizing of the stakeholders, so it was tailored to aim to assist and empower are diverse and interconnected:

Filipino Civil Society and Individuals. The need for accurate, timely, and accessible information on climate change is paramount. Filipino citizens, across all demographics, need a reliable platform to stay informed about the ongoing efforts and progress made in aligning the Philippines with The Paris Climate Agreement. Climate information disseminated through traditional channels often lacks the real-time and actionable qualities required for meaningful engagement. As a result, Filipinos require a complementary means of effective communication for climate-related insights.

Public Health Authorities. During the COVID-19 pandemic, public health authorities played a vital role in delivering real-time epidemiological insights to the public. A similar approach is needed for climate action. These authorities need a dedicated platform to communicate climate-related data effectively to the public, fostering informed decision-making at both individual and community levels.

Filipino Youth. Engaging the youth in climate action is essential for ensuring a sustainable future. Project K5 Progress acknowledges the vital role young Filipinos can play in driving change. By providing them with a platform for involvement, we aim to ignite their passion for environmental stewardship and encourage their active participation in shaping a resilient future for their country.

DEFINE

Climate change is a global challenge, and information is a critical resource in addressing it. However, existing information systems often fall short in providing local timely, relevant, and localized climate data to the Filipino populace. Much like the COVID-19 pandemic emphasized the importance of reliable information in making health-informed decisions [9], the climate crisis underscores the need for accurate, accessible, and up-to-date climate-related information also to galvanize every Filipinos. Nevertheless, the internet and social media, while powerful tools for disseminating information, also contribute to the proliferation of climate misinformation, creating an urgent need for a trusted and comprehensive climate information system.

IDEATE

In response to these challenges, the ideation of the Project K5 Progress seeks to harness the power of technology to create a web-based application that delivers climate insights to Filipinos. Inspired by the success of COVID-19 dashboards, such as the one developed by Johns Hopkins University as well as the Our World in Data foundation [10], [11], Project K5 Progress will a web application that offers a comprehensive view of climate data that is essential.

Unlike existing climate platforms, the web application will prioritize local inclusivity, ensuring that it incorporates granular insights specific to the Philippines, including the progress made in alignment with The Paris Agreement.

By leveraging innovative technology and automation, we aim to provide a robust climate information system that empowers Filipinos to make informed decisions, fosters collaboration, and engages the youth in climate action. By offering a multifunctional web application that empowers Filipinos to access, report, engage, and educate, Project K5 Progress is poised to drive climate action at the grassroots level, creating a more sustainable and resilient future for the Philippines.

Project K5 Progress will leverage technology to create a dynamic web application that delivers a host of climate-related information and context. The project will offer a multifaceted approach to catalyzing climate action:

Real-time Climate Data. The web app will provide users with access to real-time climate data and context, specific to the Philippines. This data will be updated continuously to ensure accuracy and relevance.

Climate Reporting. Users, including local governments and environmental agencies, will have the ability to report climate-related events, initiatives, and progress. This reporting feature will enhance transparency and collaboration in the climate action community.

Community Engagement. An integrated forum and discussion platform will foster community engagement and knowledge sharing. Users can participate in discussions, share best practices, and collaborate on local climate projects.

Youth Empowerment. A dedicated section of the web app will be designed to engage and educate young Filipinos about climate change. Interactive educational materials, challenges, and events will inspire and empower the youth to take meaningful climate action.

Global Climate Commitments. The most unique and salient feature is web app will keep users informed about the Philippines' progress in meeting global climate commitments, including milestones under The Paris Agreement. This will promote awareness and accountability.

OBJECTIVES

The objectives of Project K5 Progress are clear and well-defined. The first objective is to develop a user-friendly web application that serves as a hub for monitoring, reporting, and promoting progress in the Philippines' efforts to align with The Paris Agreement. This application will be designed to provide users with accessible and accurate climate data that is specifically relevant to the Philippines. It will also allow users to track the country's progress in meeting its climate goals [6]. Additionally, the platform will enable the target users to access reports on local climate initiatives and their progress with the local community.

The second objective is to provide accurate climate data and updates that inform decision-making. To achieve this, Project K5 Progress will source data from reputable sources, including meteorological agencies, scientific research institutions, and international climate databases. The project will employ data visualization techniques to present complex information in a user-friendly and understandable manner, ensuring that users can make informed decisions regarding climate action.

The third objective is to empower and engage local Filipino individuals, especially the youth, and communities in climate action. This will be achieved through various features integrated into the web application.

Users will have the opportunity to find like-minded individuals and organizations, fostering a sense of community and collaboration. The platform will also provide educational resources, enabling users to deepen their understanding of climate change and its impacts. Furthermore, the project will actively emphasize the involvement of youth, recognizing their crucial role in shaping a sustainable and resilient future. Winning the race against the climate crisis.

PROTOTYPE

For the project rationale to be conceivable, an application tech stack must be chosen properly for it to be feasible, measurable, attainable, and cost-efficient.

Fortunately, this can be achieved by following a systematic procedure to ensure that everything from the prototyping to the implementation of the Project K5 system design is achieved.

Envisioned Project Development Stages

The development of the K5 Project will utilize an Agile model within the Software Development Life Cycle (SDLC). While the most basic SDLC model commonly used is the Waterfall approach for software and web development, attempting to employ the traditional Waterfall model in a real-world web application development project is unfeasible due to its idealistic and challenging implementation [12].

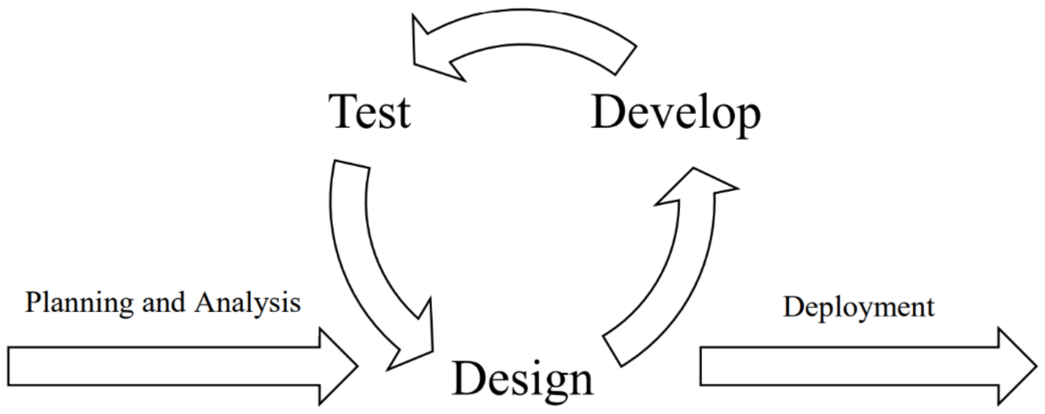


Figure 1. Agile Software Development Life Cycle Model

Additionally, the sequential nature of the Waterfall SDLC renders it unsuitable for this project since, for instance, a problem may arise during the development process, and another solution might be implemented and revise the previous stages of the cycle. Thus, the project's development methodology will embrace the Agile Model, which significantly differs from the anticipated linear and sequential approach of the Waterfall Model. The project development will benefit from this model due to its flexibility.

The primary objective of employing the Agile Software Development model is to enable swift project completion while remaining adaptable. Agile's distinctive characteristic, as illustrated in Figure 1, lies in its iterative and incremental approach, which permits developers to adjust to unforeseen circumstances throughout the development process.

The developer can refine and adapt their project as it progresses when using the Agile Model. Conversely, the Waterfall SDLC model requires the stakeholder to plan and structure everything in advance of project initiation. However, due to its rigid, linear, and sequential nature, any errors or issues that arise may go unaddressed and unresolved, as noted by Chandra [13].

For these reasons, the project lead has chosen to embrace the Agile Model as it aligns best with the project's needs. This SDLC model offers numerous advantages, including adaptability, efficiency, flexibility, continuous iteration, a high success rate with reduced time requirements, risk mitigation, and cost savings. Therefore, the envisioned project stages will be executed using the Agile Model.

Planning and Preliminary Analysis Phase

The initial phase of planning and analysis serves as the groundwork before actual development begins. During this stage, the researcher will focus on assessing the prerequisites for the COVID Pulse web application's development to ensure it aligns with the project's objectives. An essential aspect of this stage is conducting a Feasibility Analysis, where various factors, including Technical, Economic, Legal, and Scheduling (TELOS) considerations, are evaluated [14]. Additionally, this phase involves addressing potential conflicts and repercussions within the development process. It also enables the project to document, evaluate, validate, and maintain the development procedure's protocol.

TECHNICAL APPROACH

The technical approach of Project K5 Progress is built on a solid theoretical foundation and a well-structured methodology. The theoretical background encompasses several key elements. Firstly, the project is firmly grounded in the principles and goals of The Paris Agreement, ensuring alignment with international climate commitments. Secondly, it draws on climate science and data analysis methodologies to handle and present climate data accurately. Thirdly, the project leverages user-centered design principles, ensuring that the web application is intuitive and accessible to a wide range of users. Lastly, it recognizes the significance of community engagement in addressing climate change, underpinning the importance of collaboration. The method and design of the project involve several key steps. Firstly, the development of the web application is a central component. This application will be the primary interface for users to access climate data, monitor progress, and engage with the community. It will be designed with user-friendliness in mind, making it accessible even to individuals with limited technical expertise.

Data acquisition is a critical step in ensuring the reliability of the information presented on the platform. The project will source climate data and Philippine progress from established and reputable sources, including government agencies and scientific institutions. Moreover, Data analysis methods will be applied to validate and interpret the data. This may involve statistical analysis and data visualization techniques to make complex information more digestible for users. Lastly, community engagement is an integral aspect of the technical approach. The web application will feature tools and mechanisms to facilitate user interaction and collaboration. Additionally, the project will provide educational content, equipping users with the knowledge and resources they need to actively participate in climate initiatives. The involvement of youth will be a priority, recognizing their potential to drive meaningful change and advocate for a sustainable future.

TECHNICAL STACK

The selection of our technical stack is driven by the imperative of time constraint. There is a need to ensure the timely development and deployment of a functional web application. Therefore, the project lead has chosen technologies that strike a balance between efficiency, reliability, and ease of implementation.

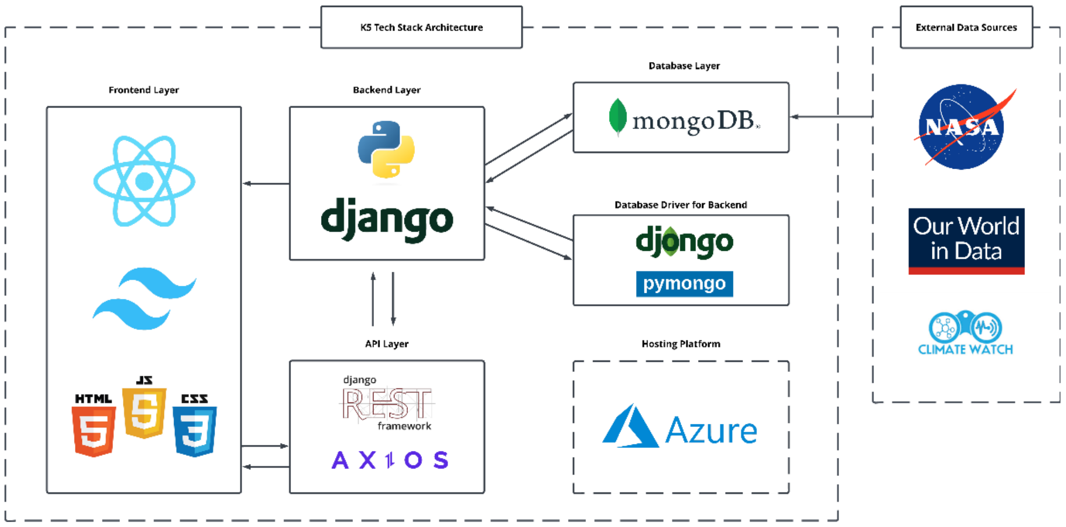


Figure 2. K5 Tech Stack Architecture

PLANNING AND ANALYSIS STAGE

The planning and analysis stage of Project K5 Progress represents the initial phase before the actual development efforts begin. In this phase, the focus is on evaluating the essential prerequisites for the successful development of the K5 Progress web application, ensuring that it aligns seamlessly with our project's core objectives. Central to this phase is the conduct of a Feasibility Analysis, encompassing a comprehensive assessment of various critical factors, including Technical, Economic, Legal, and Scheduling (TELOS) considerations, as detailed by McLeod (2021). This phase also serves as a platform for addressing potential conflicts and consequences that may arise during the development process while facilitating the documentation, assessment, verification, and maintenance of the development protocol.

In the preliminary development stage, Project K5 Progress requires a substantial overhaul, including web application enhancement and the implementation of responsive web design principles. At this juncture, our primary focus will be on assessing technical, operational, and scheduling feasibility based on the TELOS framework [14]. Economic, human resource, and legal factors will not be within the scope of this assessment, as they are either inconclusive or deemed unnecessary for the straightforward nature of this web application project. Additionally, it's worth noting that the project's financial requirements are minimal due to the accessibility of most resources, with many being open source. The researcher has already conducted a preliminary evaluation of the technical resources required and their suitability for meeting the developmental needs of Project K5 Progress.

Regarding hardware prerequisites, aside from a standard computer and internet connection, there is no substantial need for additional hardware specifications. This is because our intention is to leverage the cloud-based service, which is tentatively chosen as Azure, to host the K5 Progress web application, obviating the need for dedicated server hardware.

Furthermore, it is essential to underscore that Project K5 Progress is not only technically viable but also built upon a well-established concept. The requisite technology for the development of this web application is readily available, with most of the necessary tools being open source (i.e., Django, React, REST Framework, PyMongo, Djongo). The researcher possesses substantial expertise in several key technical domains and holds a Responsive Web Design certificate (Larson, 2021) to validate their competency.

In scenarios where unforeseen technical requirements may exceed the development team’s current expertise, flexible and adaptive solutions can be sought to fulfill those specific needs. Consequently, considering these factors, the technical feasibility of implementing Project K5 Progress is highly promising. Within the TELOS framework, our project will place the most substantial emphasis on assessing technical feasibility.

To this end, the following software components are deemed essential for its successful development: The choice JavaScript and React for the frontend, along with Axios for data exchange, provides a solid foundation for responsive web design and efficient UI development. React's component-based structure enhances development efficiency, which is crucial for meeting project timelines. For the backend, Python and Django are chosen for their simplicity and high-level capabilities. Django's built-in functionalities and the Django REST Framework for API development align perfectly with the need to establish robust data exchange mechanisms swiftly. MongoDB, along with Djongo and PyMongo, serves as the database and API stack. MongoDB's agility in managing unstructured or semi-structured data is well-suited for the project's data requirements. Djongo bridges Django and MongoDB, allowing you to leverage Django's ORM capabilities while using MongoDB as the database system, reducing development time associated with data management.

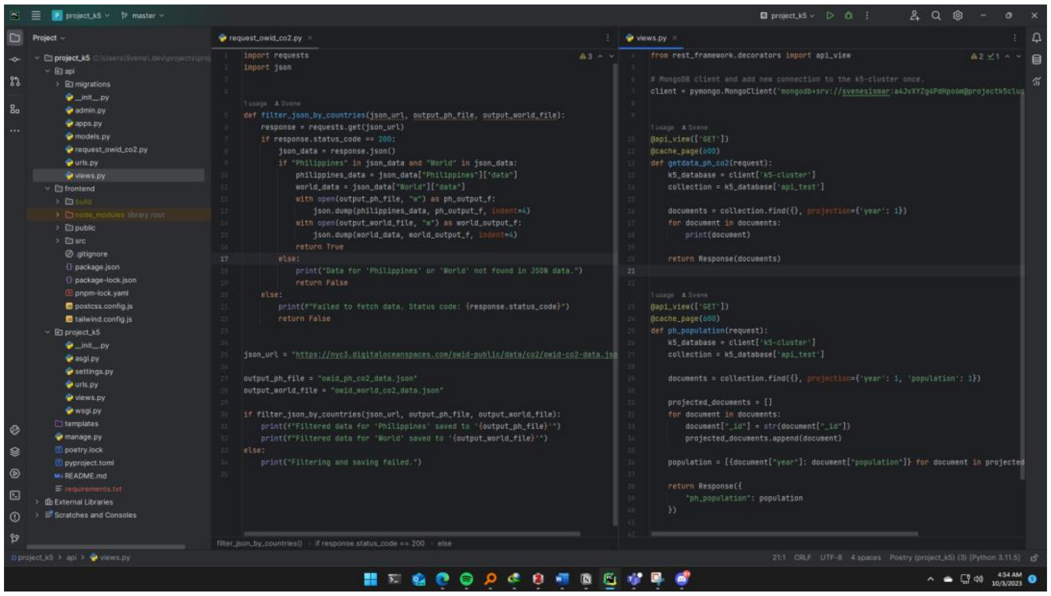
Scheduling and Operational Feasibility. The chosen tech stack is well-documented and has a strong community, ensuring access to resources and support for operational purposes. This enhances the operational feasibility of your project. The modularity of React and the pre-built tools in Django and Django REST Framework facilitate faster development. Tailwind CSS expedites UI styling and customization, which is critical for meeting project time constraints while maintaining design quality. These factors contribute to the scheduling feasibility of your project.

PROOF OF CONCEPT

Proof of concept (PoC) plays a pivotal role in Project K5's development journey, serving as a critical milestone. By creating a PoC, the project can validate the feasibility of its core ideas and technologies before committing to full-scale development. This preliminary phase allows for the identification of potential challenges, the testing of integration between Django, React, and MongoDB, and the assessment of performance optimization strategies.

Additionally, the PoC provides a valuable opportunity to showcase the project's capabilities for the current project proposal, gaining confidence and support. Overall, a well-executed proof of concept is indispensable in ensuring that Project K5's development proceeds smoothly, efficiently, and with a clear understanding of its technical viability and potential for success.

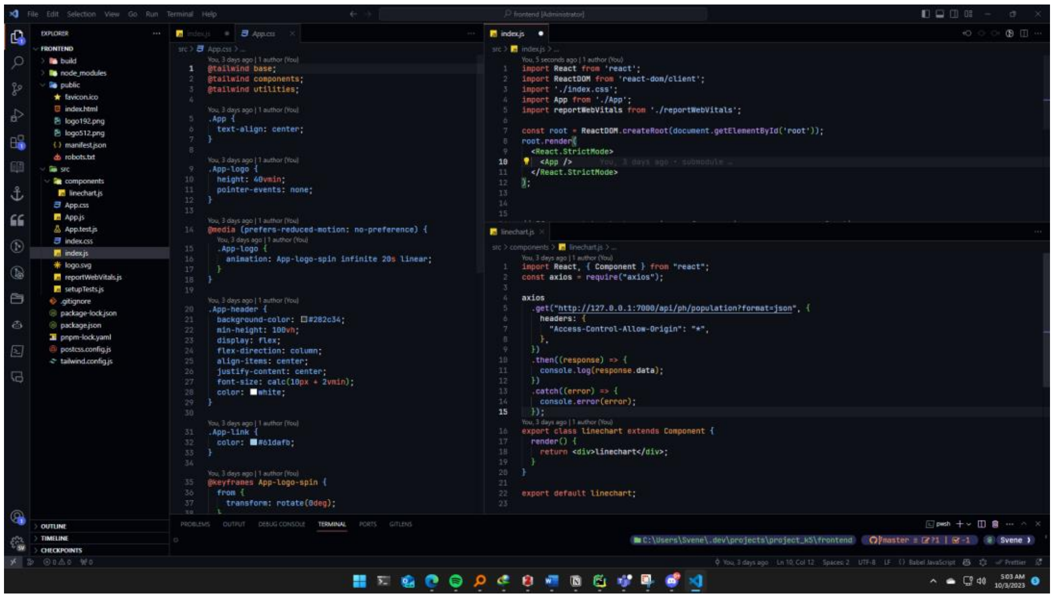
After meticulously validating the proof of concept, it is evident that the technical feasibility of our project has been confirmed. It has successfully achieved key milestones across multiple project layers, including Backend, Frontend, Database, and Application Programming Interface (API). In the following sections, the brief documentation will provide detailed insights into each component's setup and integration.



```python
import requests
import json


def filter_json_by_countries(json_url, output_ph_file, output_world_file):
    response = requests.get(json_url)
    if response.status_code == 200:
        json_data = response.json()
        if "Philippines" in json_data and "World" in json_data:
            philippines_data = json_data["Philippines"]["data"]
            world_data = json_data["World"]["data"]
            with open(output_ph_file, "w") as ph_output_f:
                json.dump(philippines_data, ph_output_f, indent=4)
            with open(output_world_file, "w") as world_output_f:
                json.dump(world_data, world_output_f, indent=4)
            return True
        else:
            print("Data for 'Philippines' or 'World' not found in JSON data.")
            return False
    else:
        print(f"Failed to fetch data. Status code: {response.status_code}")
        return False


json_url = "https://nyc3.digitaloceanspaces.com/owid-public/data/co2/owid-co2-data.jso

output_ph_file = "owid_ph_co2_data.json"
output_world_file = "owid_world_co2_data.json"

if filter_json_by_countries(json_url, output_ph_file, output_world_file):
    print(f"Filtered data for 'Philippines' saved to '{output_ph_file}'")
    print(f"Filtered data for 'World' saved to '{output_world_file}'")
else:
    print("Filtering and saving failed.")
```

```python
from rest_framework.decorators import api_view

# MongoDB client and add new connection to the k5-cluster once.
client = pymongo.MongoClient('mongodb+srv://svenesismar:a4JvXYZg4PdMpo6m@projectk5clus


@api_view(['GET'])
@cache_page(600)
def getdata_ph_co2(request):
    k5_database = client['k5-cluster']
    collection = k5_database['api_test']

    documents = collection.find({}, projection={'year': 1})
    for document in documents:
        print(document)

    return Response(documents)


@api_view(['GET'])
@cache_page(600)
def ph_population(request):
    k5_database = client['k5-cluster']
    collection = k5_database['api_test']

    documents = collection.find({}, projection={'year': 1, 'population': 1})

    projected_documents = []
    for document in documents:
        document["_id"] = str(document["_id"])
        projected_documents.append(document)

    population = [{document["year"]: document["population"]} for document in projected

    return Response({
        "ph_population": population
    })
```

Figure 3. *K5 Project Backend in PyCharm*



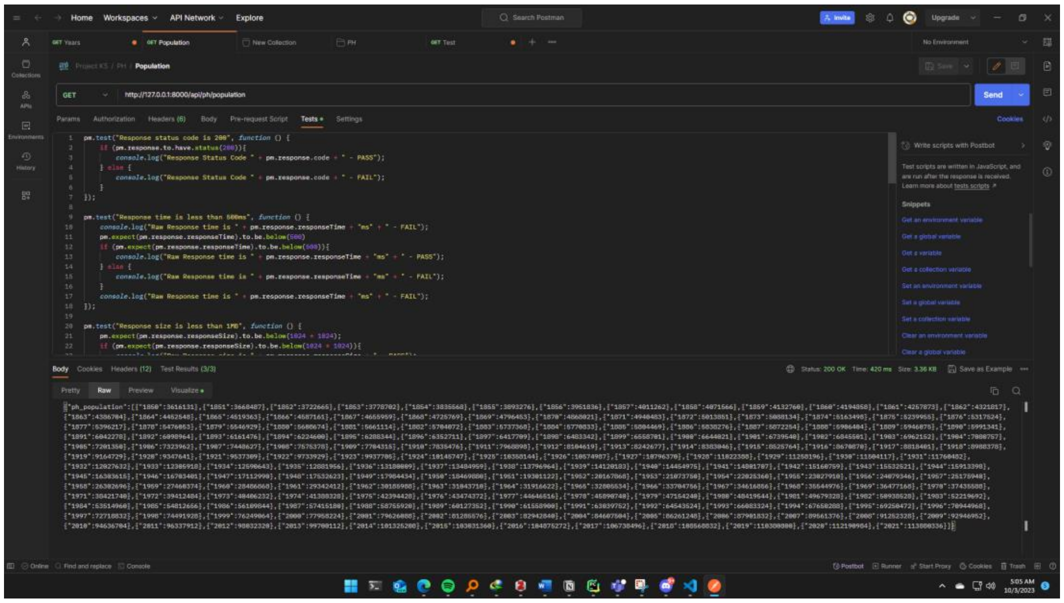
```
src > App.css
@tailwind base;
@tailwind components;
@tailwind utilities;

.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
```

```
src > index.js
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

```
src > components > linechart.js
import React, { Component } from "react";
const axios = require("axios");

axios
  .get("http://127.0.0.1:7000/api/ph/population?format=json", {
    headers: {
      "Access-Control-Allow-Origin": "*",
    },
  })
  .then((response) => {
    console.log(response.data);
  })
  .catch((error) => {
    console.error(error);
  });
export class linechart extends Component {
  render() {
    return <div>linechart</div>;
  }
}

export default linechart;
```

```
C:\Users\Svene\.dev\projects\project_k5\frontend
```

Figure 4. K5 Project Frontend in VSCode



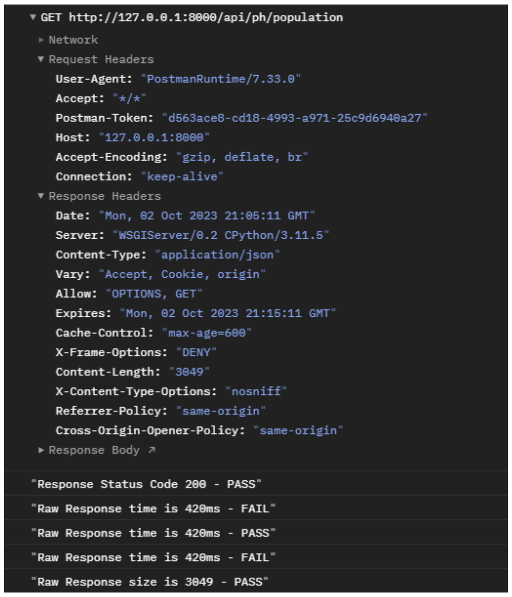
```
Project K5 / PH / Population
GET http://127.0.0.1:8000/api/ph/population

pm.test("Response status code is 200", function () {
    if (pm.response.to.have.status(200)){
        console.log("Response Status Code " + pm.response.code + " - PASS");
    } else {
        console.log("Response Status Code " + pm.response.code + " - FAIL");
    }
});

pm.test("Response time is less than 500ms", function () {
    console.log("Raw Response time is " + pm.response.responseTime + "ms" + " - FAIL");
    pm.expect(pm.response.responseTime).to.be.below(500)
    if (pm.expect(pm.response.responseTime).to.be.below(500)){
        console.log("Raw Response time is " + pm.response.responseTime + "ms" + " - PASS");
    } else {
        console.log("Raw Response time is " + pm.response.responseTime + "ms" + " - FAIL");
    }
    console.log("Raw Response time is " + pm.response.responseTime + "ms" + " - FAIL");
});

pm.test("Response size is less than 1MB", function () {
    pm.expect(pm.response.responseSize).to.be.below(1024 * 1024);
    if (pm.expect(pm.response.responseSize).to.be.below(1024 * 1024)){
```

```
Body  Cookies  Headers (12)  Test Results (3/3)      Status: 200 OK  Time: 420 ms  Size: 3.36 KB
```

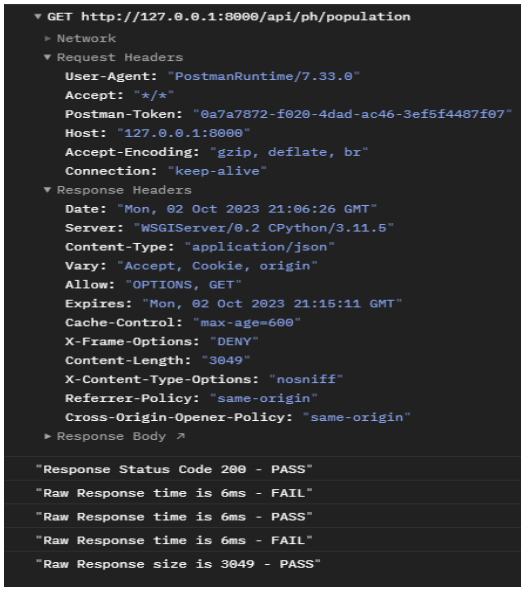
```
{"ph_population":[{"1850":3616131},{"1851":3668487},{"1852":3722665},{"1853":3778702},{"1854":3835568},{"1855":3893276},{"1856":3951836},{"1857":4011262},{"1858":4071566},{"1859":4132760},{"1860":4194858},{"1861":4257873},{"1862":4321817},
{"1863":4386704},{"1864":4452548},{"1865":4519363},{"1866":4587161},{"1867":4655959},{"1868":4725769},{"1869":4796463},{"1870":4868021},{"1871":4940483},{"1872":5013851},{"1873":5088134},{"1874":5163498},{"1875":5239965},{"1876":5317524},
{"1877":5396217},{"1878":5476063},{"1879":5546929},{"1880":5600674},{"1881":5661114},{"1882":5704072},{"1883":5737360},{"1884":5770833},{"1885":5804469},{"1886":5838276},{"1887":5872254},{"1888":5906404},{"1889":5940675},{"1890":5991341},
{"1891":6042278},{"1892":6098964},{"1893":6161476},{"1894":6224600},{"1895":6288344},{"1896":6352711},{"1897":6417709},{"1898":6483342},{"1899":6550701},{"1900":6644821},{"1901":6739540},{"1902":6845501},{"1903":6962152},{"1904":7080757},
{"1905":7201350},{"1906":7323962},{"1907":7448627},{"1908":7575378},{"1909":7704315},{"1910":7835476},{"1911":7968898},{"1912":8104619},{"1913":8242677},{"1914":8383046},{"1915":8525764},{"1916":8670870},{"1917":8818401},{"1918":8968378},
{"1919":9164729},{"1920":9347641},{"1921":9537309},{"1922":9733929},{"1923":9937705},{"1924":10146747},{"1925":10360144},{"1926":10574987},{"1927":10796370},{"1928":11022300},{"1929":11260196},{"1930":11504117},{"1931":11760482},
{"1932":12027632},{"1933":12305910},{"1934":12590643},{"1935":12881956},{"1936":13180889},{"1937":13484959},{"1938":13796964},{"1939":14120183},{"1940":14454975},{"1941":14801707},{"1942":15160759},{"1943":15532521},{"1944":15913398},
{"1945":16303615},{"1946":16703401},{"1947":17112990},{"1948":17532623},{"1949":17964434},{"1950":18409880},{"1951":19301122},{"1952":20167868},{"1953":21073750},{"1954":22025360},{"1955":23027910},{"1956":24079346},{"1957":25175948},
{"1958":26302696},{"1959":27460374},{"1960":28486060},{"1961":29342412},{"1962":30185908},{"1963":31043710},{"1964":31916622},{"1965":32805534},{"1966":33704756},{"1967":34616056},{"1968":35544976},{"1969":36477168},{"1970":37435580},
{"1971":38421740},{"1972":39412484},{"1973":40406232},{"1974":41388328},{"1975":42394428},{"1976":43474372},{"1977":44646516},{"1978":45890748},{"1979":47154248},{"1980":48419544},{"1981":49679328},{"1982":50938528},{"1983":52219692},
{"1984":53514960},{"1985":54812666},{"1986":56109844},{"1987":57415188},{"1988":58755920},{"1989":60127352},{"1990":61558900},{"1991":63039752},{"1992":64543524},{"1993":66083324},{"1994":67658288},{"1995":69250472},{"1996":70944968},
{"1997":72718832},{"1998":74491920},{"1999":76249064},{"2000":77958224},{"2001":79626088},{"2002":81285676},{"2003":82942840},{"2004":84607504},{"2005":86261248},{"2006":87901832},{"2007":89561376},{"2008":91252328},{"2009":92946952},
{"2010":94636704},{"2011":96337912},{"2012":98032320},{"2013":99700112},{"2014":101325200},{"2015":103031360},{"2016":104875272},{"2017":106738496},{"2018":108568832},{"2019":110380880},{"2020":112190904},{"2021":113880336}]}
```

Figure 5. K5 Project API in Postman

Proof of Concept API



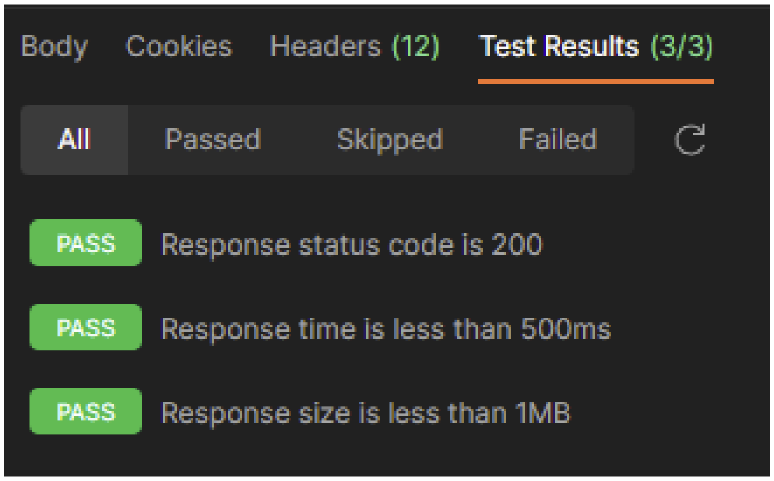
```
▾ GET http://127.0.0.1:8000/api/ph/population
  ▸ Network
  ▾ Request Headers
    User-Agent: "PostmanRuntime/7.33.0"
    Accept: "*/*"
    Postman-Token: "d563ace8-cd18-4993-a971-25c9d6940a27"
    Host: "127.0.0.1:8000"
    Accept-Encoding: "gzip, deflate, br"
    Connection: "keep-alive"
  ▾ Response Headers
    Date: "Mon, 02 Oct 2023 21:05:11 GMT"
    Server: "WSGIServer/0.2 CPython/3.11.5"
    Content-Type: "application/json"
    Vary: "Accept, Cookie, origin"
    Allow: "OPTIONS, GET"
    Expires: "Mon, 02 Oct 2023 21:15:11 GMT"
    Cache-Control: "max-age=600"
    X-Frame-Options: "DENY"
    Content-Length: "3049"
    X-Content-Type-Options: "nosniff"
    Referrer-Policy: "same-origin"
    Cross-Origin-Opener-Policy: "same-origin"
  ▸ Response Body ↗
"Response Status Code 200 - PASS"
"Raw Response time is 420ms - FAIL"
"Raw Response time is 420ms - PASS"
"Raw Response time is 420ms - FAIL"
"Raw Response size is 3049 - PASS"
```



```
▾ GET http://127.0.0.1:8000/api/ph/population
  ▸ Network
  ▾ Request Headers
    User-Agent: "PostmanRuntime/7.33.0"
    Accept: "*/*"
    Postman-Token: "0a7a7872-f020-4dad-ac46-3ef5f4487f07"
    Host: "127.0.0.1:8000"
    Accept-Encoding: "gzip, deflate, br"
    Connection: "keep-alive"
  ▾ Response Headers
    Date: "Mon, 02 Oct 2023 21:06:26 GMT"
    Server: "WSGIServer/0.2 CPython/3.11.5"
    Content-Type: "application/json"
    Vary: "Accept, Cookie, origin"
    Allow: "OPTIONS, GET"
    Expires: "Mon, 02 Oct 2023 21:15:11 GMT"
    Cache-Control: "max-age=600"
    X-Frame-Options: "DENY"
    Content-Length: "3049"
    X-Content-Type-Options: "nosniff"
    Referrer-Policy: "same-origin"
    Cross-Origin-Opener-Policy: "same-origin"
  ▸ Response Body ↗
"Response Status Code 200 - PASS"
"Raw Response time is 6ms - FAIL"
"Raw Response time is 6ms - PASS"
"Raw Response time is 6ms - FAIL"
"Raw Response size is 3049 - PASS"
```

Observations in the Default Implementation

Pymongo driver was used, which is not asynchronous. For better performance, the `pymongo-async` driver was opted to be used. Another reason why the default implementation was problematic was because a new MongoDB client was created and connecting to the database in each view. This can be slow, especially if we have a lot of traffic to the API.



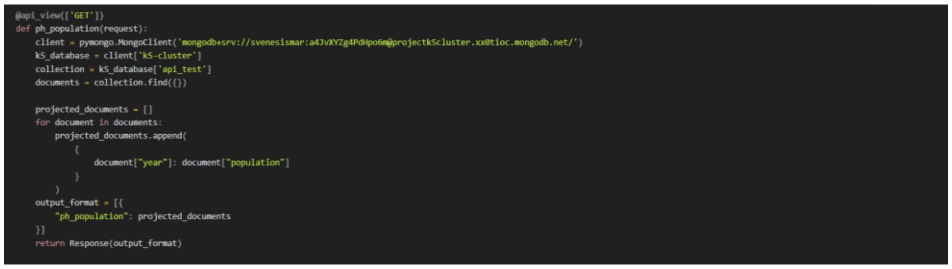
PASS Response status code is 200

PASS Response time is less than 500ms

PASS Response size is less than 1MB

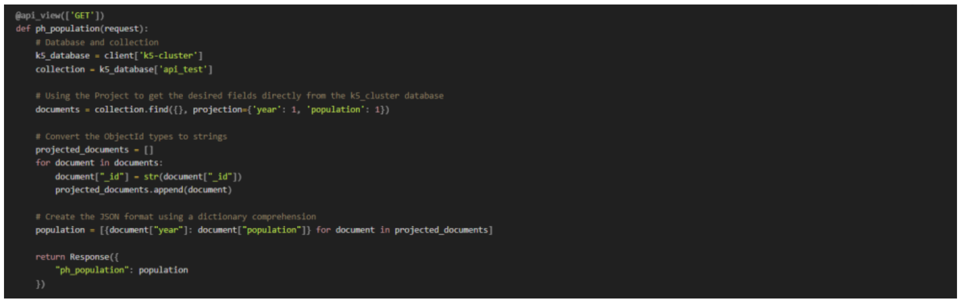
It would be more efficient to create a single client and connect to the database once, at the start of your application. Lastly, the default configuration was iterating over the results of the `collection.find()` query in each view and creating a new list of documents to return. This can be slow, especially if the query returns a lot of results. It would be more efficient to use the `$project` operator to project the desired fields directly from the database. Then the cache_page decorator was implemented, which was very crucial. This decorator will automatically cache the view's response, for the specified timeout period. This literally presented significant improvements to the performance of your REST Framework API.

Before the Improvement Implementation

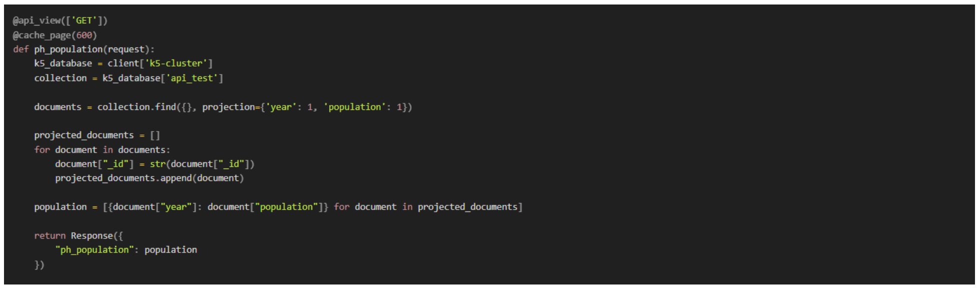


```
@api_view(['GET'])
def ph_population(request):
    client = pymongo.MongoClient('mongodb+srv://svenesismar:a4JvXYZg4PdHpo6m@projectk5cluster.xx0tioc.mongodb.net/')
    k5_database = client['k5-cluster']
    collection = k5_database['api_test']
    documents = collection.find({})

    projected_documents = []
    for document in documents:
        projected_documents.append(
            {
                document["year"]: document["population"]
            }
        )
    output_format = [{
        "ph_population": projected_documents
    }]
    return Response(output_format)
```

After the Improvement Implementation

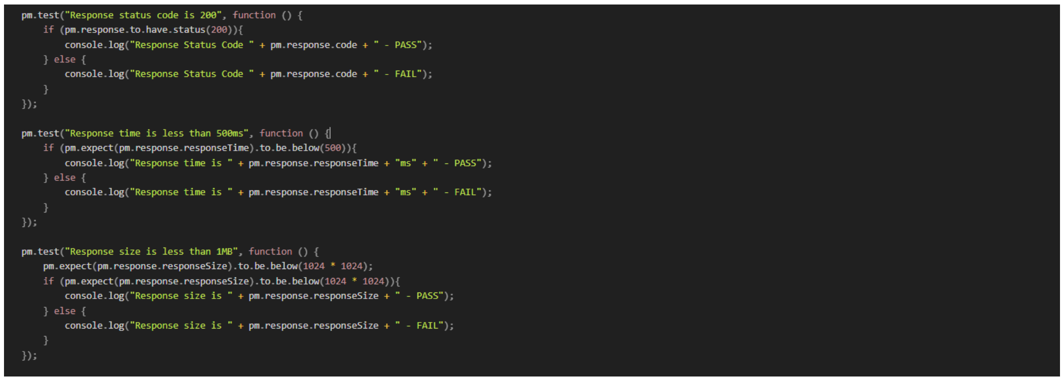


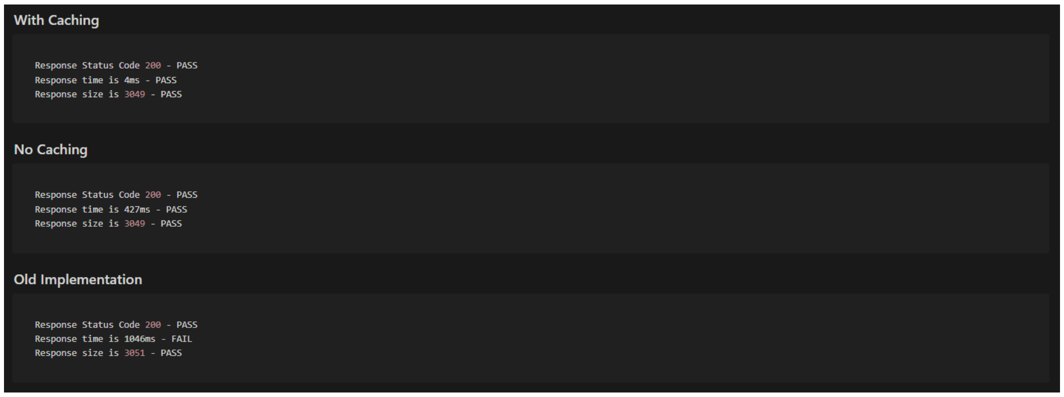
```
@api_view(['GET'])
def ph_population(request):
    # Database and collection
    k5_database = client['k5-cluster']
    collection = k5_database['api_test']

    # Using the Project to get the desired fields directly from the k5_cluster database
    documents = collection.find({}, projection={'year': 1, 'population': 1})

    # Convert the ObjectId types to strings
    projected_documents = []
    for document in documents:
        document["_id"] = str(document["_id"])
        projected_documents.append(document)

    # Create the JSON format using a dictionary comprehension
    population = [{document["year"]: document["population"]} for document in projected_documents]

    return Response({
        "ph_population": population
    })
```



```
@api_view(['GET'])
@cache_page(600)
def ph_population(request):
    k5_database = client['k5-cluster']
    collection = k5_database['api_test']

    documents = collection.find({}, projection={'year': 1, 'population': 1})

    projected_documents = []
    for document in documents:
        document["_id"] = str(document["_id"])
        projected_documents.append(document)

    population = [{document["year"]: document["population"]} for document in projected_documents]

    return Response({
        "ph_population": population
    })
```

Postman API Performance Testing


```
pm.test("Response status code is 200", function () {
    if (pm.response.to.have.status(200)){
        console.log("Response Status Code " + pm.response.code + " - PASS");
    } else {
        console.log("Response Status Code " + pm.response.code + " - FAIL");
    }
});

pm.test("Response time is less than 500ms", function () {
    if (pm.expect(pm.response.responseTime).to.be.below(500)){
        console.log("Response time is " + pm.response.responseTime + "ms" + " - PASS");
    } else {
        console.log("Response time is " + pm.response.responseTime + "ms" + " - FAIL");
    }
});

pm.test("Response size is less than 1MB", function () {
    pm.expect(pm.response.responseSize).to.be.below(1024 * 1024);
    if (pm.expect(pm.response.responseSize).to.be.below(1024 * 1024)){
        console.log("Response size is " + pm.response.responseSize + " - PASS");
    } else {
        console.log("Response size is " + pm.response.responseSize + " - FAIL");
    }
});
```


With Caching

```
Response Status Code 200 - PASS
Response time is 4ms - PASS
Response size is 3049 - PASS
```

No Caching

```
Response Status Code 200 - PASS
Response time is 427ms - PASS
Response size is 3049 - PASS
```

Old Implementation

```
Response Status Code 200 - PASS
Response time is 1046ms - FAIL
Response size is 3051 - PASS
```

Backend Setup

To test the proof of concept in the backend aspect of this project, this section will start setting up a Django Framework that involves several steps, including the installation of Django, creating a project, and starting a development server. The following is the breakdown of the process.

Creating the Project. Let us start by installing an official release with pip. This is the recommended way to install Django. The easiest is to use the standalone pip installer. If the distribution already has pip installed, there might be a need to update it if it is outdated. If it is outdated, there will be a prompt because the installation will not work.

For the virtual environment, venv, Poetry was chosen. This tool provides isolated Python environments, which are more practical than installing packages systemwide. It also allows installing packages without administrator privileges.

The contributing tutorial walks through how to create a virtual environment. However, in this instance, I will use Poetry. And after all of these are ready, the project can now be properly set up.

```
$ python -m pip install Django
```

After Django was installed, the project was created in the current directory. The following command was used to initialize the project.

```
$ django-admin startproject k5
```

This will create a folder project called ‘k5’ in the current directory.

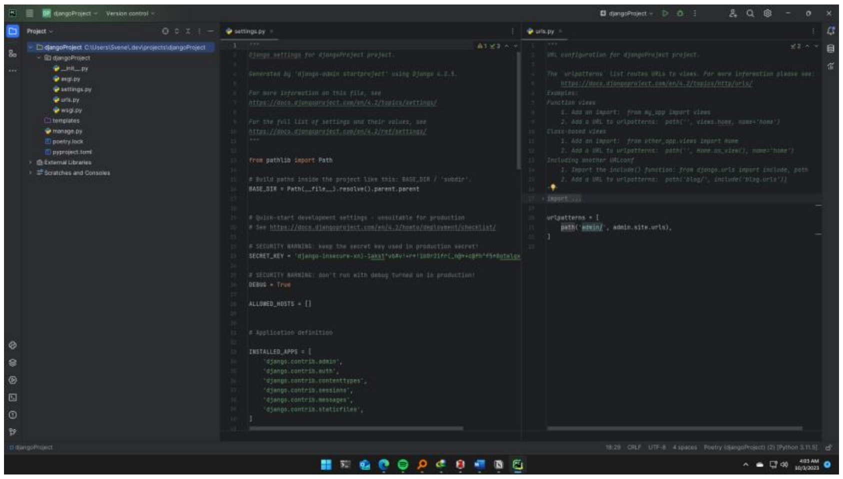
```
# Project Structure
k5/
    manage.py
    k5/
       __init__.py
       settings.py
       urls.py
       asgi.py
       wsgi.py
```

Starting the Development server. Change into the outer mysite directory, and running the following command will start the development server of the backend web application for K5:

```
$ python manage.py runserver
```

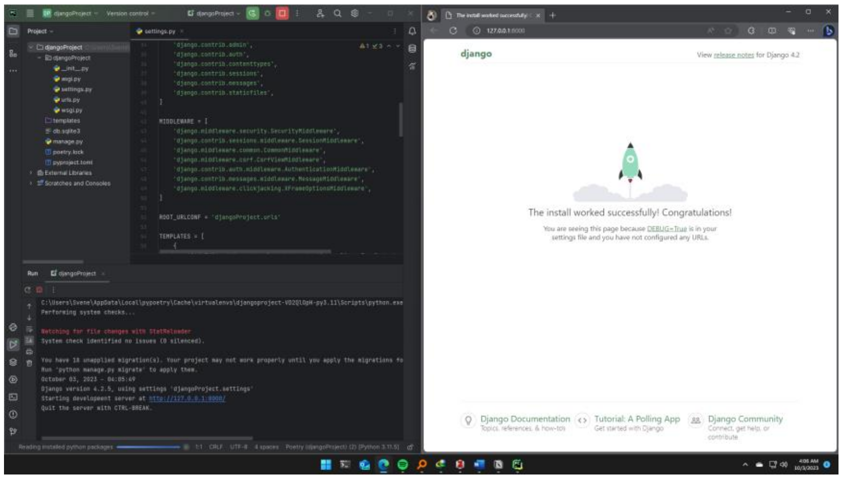
There will be an output on the command line:

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly
until you apply the migrations for app(s): admin, auth, contenttypes,
sessions.
Run 'python manage.py migrate' to apply them.
September 28, 2023 - 21:15:05
Django version 4.2.5, using settings 'k5.settings'
Starting development server at <http://127.0.0.1:8000/>
Quit the server with CTRL-BREAK.
```



```
"""
Django settings for djangoProject project.

Generated by 'django-admin startproject' using Django 4.2.5.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-xn)-1aks1*vb#v!+r+!1b0r21fr(_n@++c@#h^f5+8otelgx

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

```
"""
URL configuration for djangoProject project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
import ...

urlpatterns = [
    path('admin/', admin.site.urls),
]
```



```
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'djangoProject.urls'

TEMPLATES = [
    {
```

```
C:\Users\Svene\AppData\Local\pypoetry\Cache\virtualenvs\djangoproject-V02QlOpH-py3.11\Scripts\python.exe
Performing system checks...

Watching for file changes with StatReloader
System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations fo
Run 'python manage.py migrate' to apply them.
October 03, 2023 - 04:05:49
Django version 4.2.5, using settings 'djangoProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

django

View release notes for Django 4.2

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

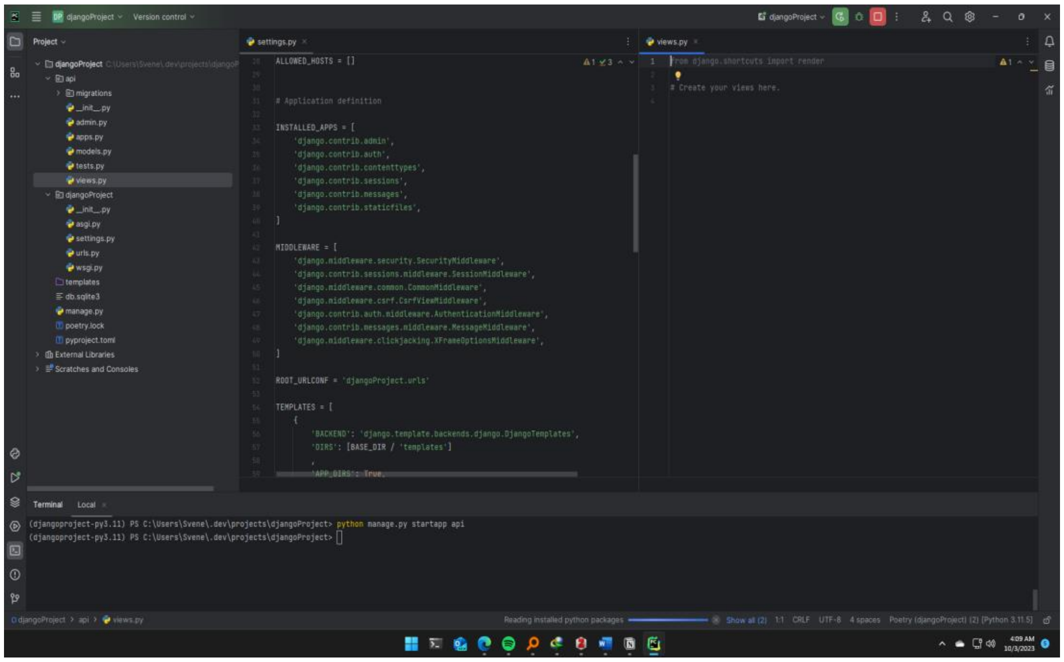
Django Documentation
Topics, references, & how-to's

Tutorial: A Polling App
Get started with Django

Django Community
Connect, get help, or contribute

Making the Rest API application in the project. The following command will create an application for the project that will handle the API-related transactions within the project.

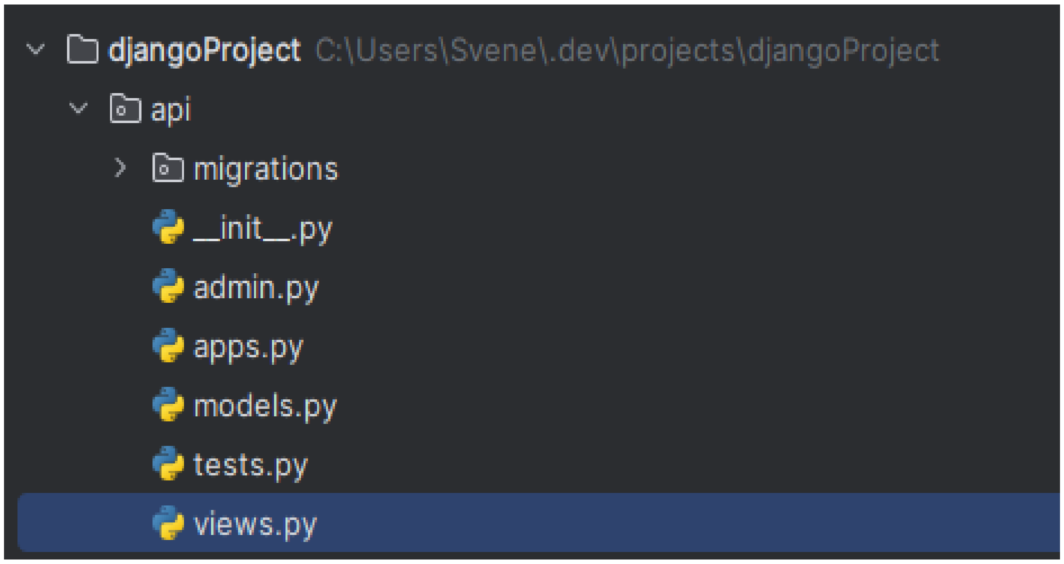
```
$ python manage.py startapp api
```

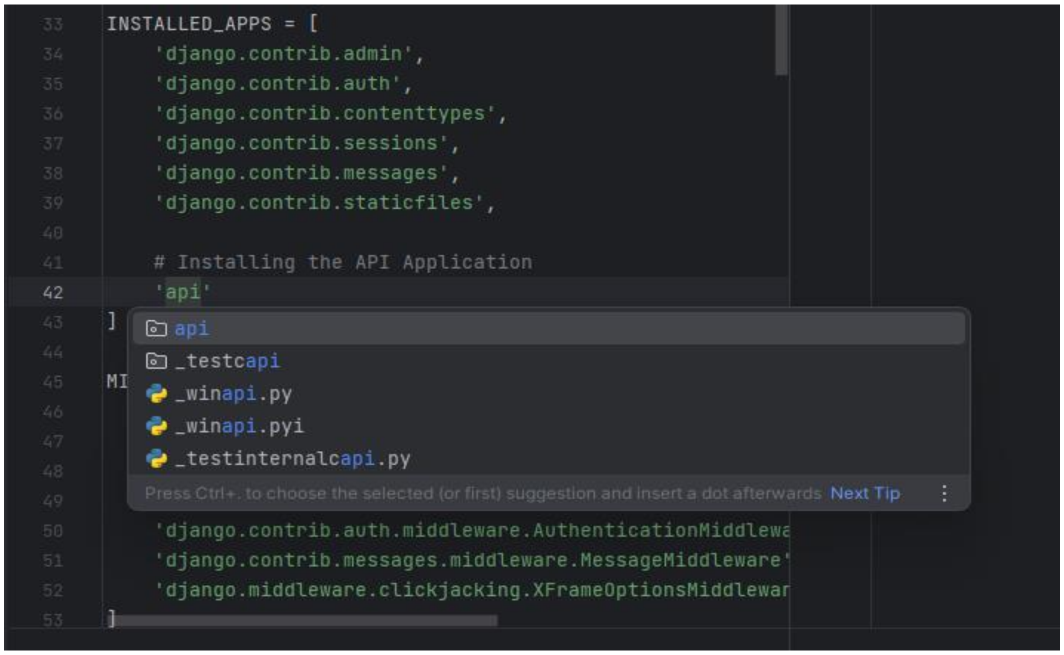


```
ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'djangoProject.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates']
        ,
        'APP_DIRS': True,
```

```
from django.shortcuts import render

# Create your views here.
```

```
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject> python manage.py startapp api
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject>
```





```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Installing the API Application
    'api'
]

MI
    'django.contrib.auth.middleware.AuthenticationMiddlewa
    'django.contrib.messages.middleware.MessageMiddleware'
    'django.middleware.clickjacking.XFrameOptionsMiddlewar
]
```

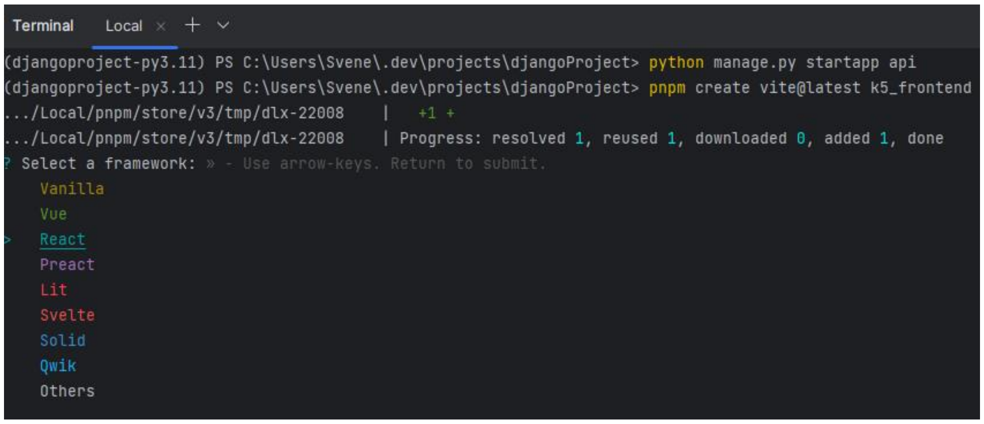
api
_testcapi
_winapi.py
_winapi.pyi
_testinternalcapi.py
Press Ctrl+. to choose the selected (or first) suggestion and insert a dot afterwards Next Tip

Frontend Setup

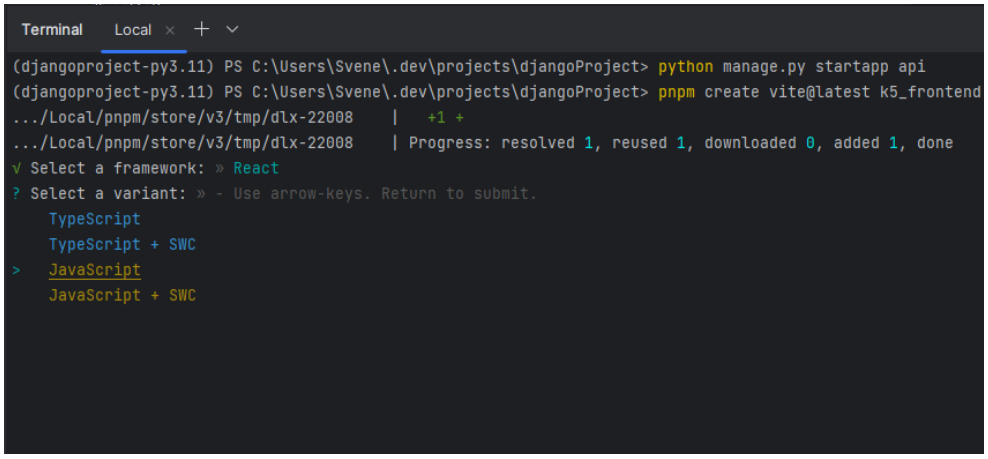
For the frontend, React is used as a frontend framework. And the same with the previous process, to test the proof of concept in the frontend aspect of this project, this section will start setting up a Django Framework that involves several steps, including the installation of Django, creating a project, and starting a development server. The following is the breakdown of the process.

Django and ReactJS Integration. To integrate Django with ReactJS, there is a need to modify the Django settings in settings.py to include the paths to the frontend templates and static files. Additionally, a view will be created in the Django app that renders the React frontend. Scaffolding the frontend with pnpm (I replaced npm due to the benefits). Then follow the prompts. This project will be using JavaScript without the SWC.

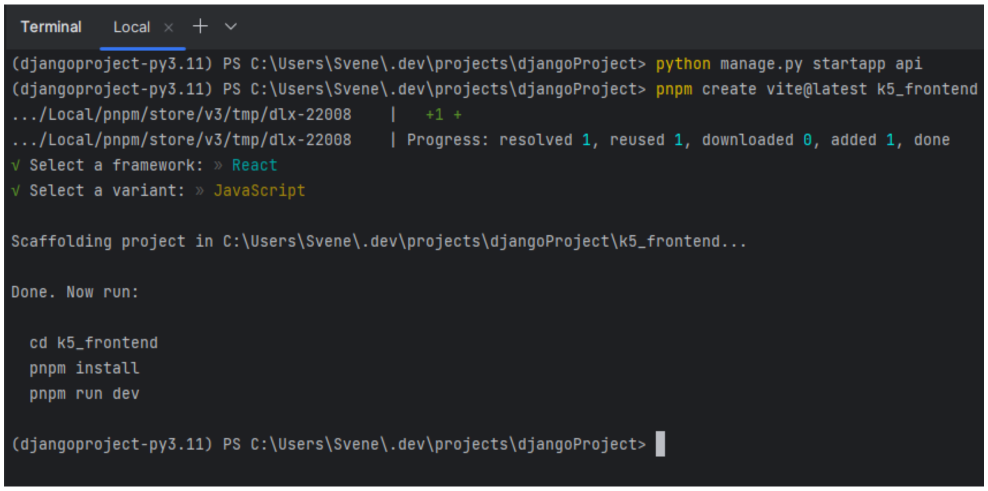
```
$ pnpm create vite@latest k5_frontend
$ cd k5_frontend
$ pnpm install
```



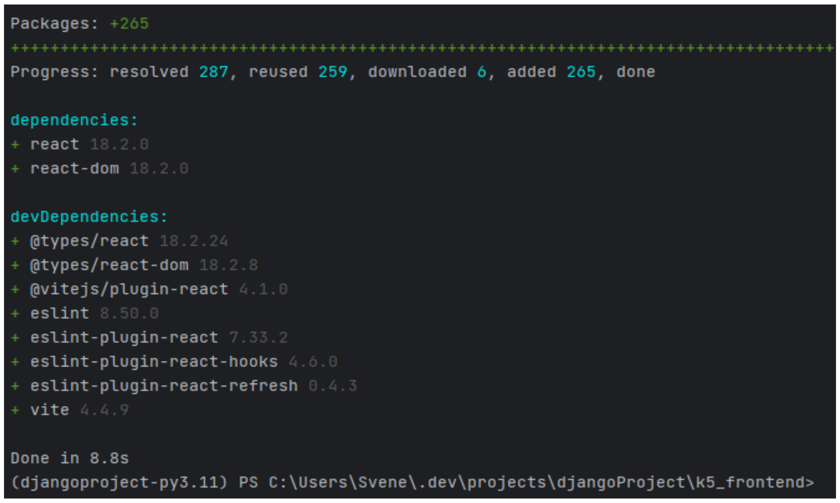
```
Terminal   Local
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject> python manage.py startapp api
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject> pnpm create vite@latest k5_frontend
.../Local/pnpm/store/v3/tmp/dlx-22008    |   +1 +
.../Local/pnpm/store/v3/tmp/dlx-22008    | Progress: resolved 1, reused 1, downloaded 0, added 1, done
? Select a framework: » - Use arrow-keys. Return to submit.
    Vanilla
    Vue
>   React
    Preact
    Lit
    Svelte
    Solid
    Qwik
    Others
```



```
Terminal   Local
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject> python manage.py startapp api
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject> pnpm create vite@latest k5_frontend
.../Local/pnpm/store/v3/tmp/dlx-22008    |   +1 +
.../Local/pnpm/store/v3/tmp/dlx-22008    | Progress: resolved 1, reused 1, downloaded 0, added 1, done
√ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
    TypeScript
    TypeScript + SWC
>   JavaScript
    JavaScript + SWC
```



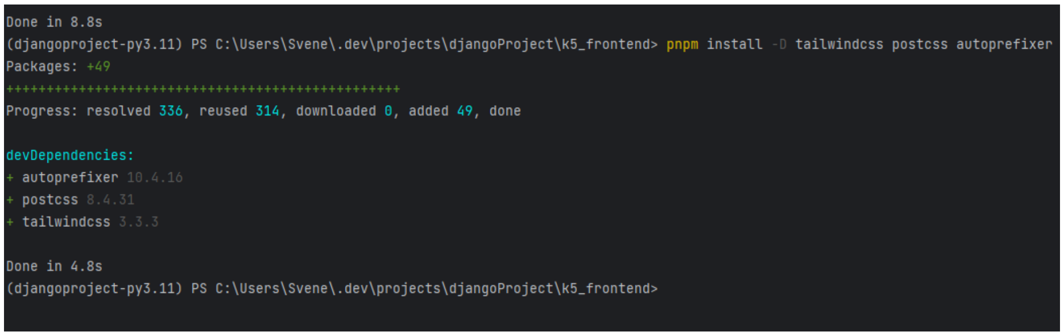
```
Terminal   Local
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject> python manage.py startapp api
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject> pnpm create vite@latest k5_frontend
.../Local/pnpm/store/v3/tmp/dlx-22008    |   +1 +
.../Local/pnpm/store/v3/tmp/dlx-22008    | Progress: resolved 1, reused 1, downloaded 0, added 1, done
√ Select a framework: » React
√ Select a variant: » JavaScript

Scaffolding project in C:\Users\Svene\.dev\projects\djangoProject\k5_frontend...

Done. Now run:

  cd k5_frontend
  pnpm install
  pnpm run dev

(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject>
```



```
Packages: +265
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Progress: resolved 287, reused 259, downloaded 6, added 265, done

dependencies:
+ react 18.2.0
+ react-dom 18.2.0

devDependencies:
+ @types/react 18.2.24
+ @types/react-dom 18.2.8
+ @vitejs/plugin-react 4.1.0
+ eslint 8.50.0
+ eslint-plugin-react 7.33.2
+ eslint-plugin-react-hooks 4.6.0
+ eslint-plugin-react-refresh 0.4.3
+ vite 4.4.9

Done in 8.8s
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject\k5_frontend>
```

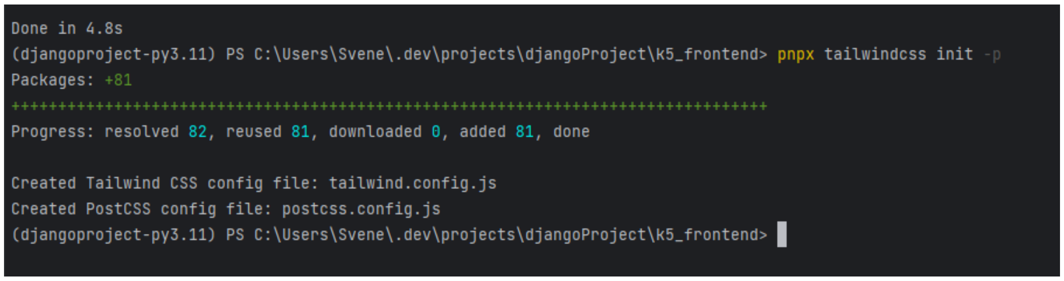
Installation of Tailwind CSS. Install tailwindcss and its peer dependencies, then generate tailwind.config.js and postcss.config.js files.

```
$ pnpm install -D tailwindcss postcss autoprefixer
```



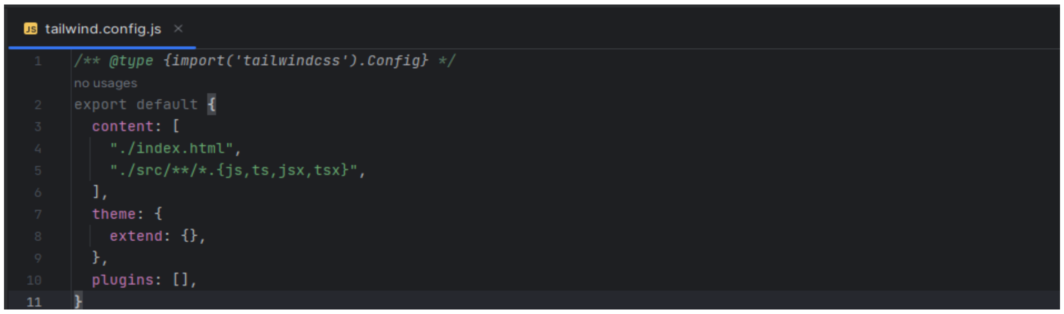
```
Done in 8.8s
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject\k5_frontend> pnpm install -D tailwindcss postcss autoprefixer
Packages: +49
+++++++++++++++++++++++++++++++++++++++++++++++++
Progress: resolved 336, reused 314, downloaded 0, added 49, done

devDependencies:
+ autoprefixer 10.4.16
+ postcss 8.4.31
+ tailwindcss 3.3.3

Done in 4.8s
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject\k5_frontend>
```

```
$ pnpx tailwindcss init -p
```



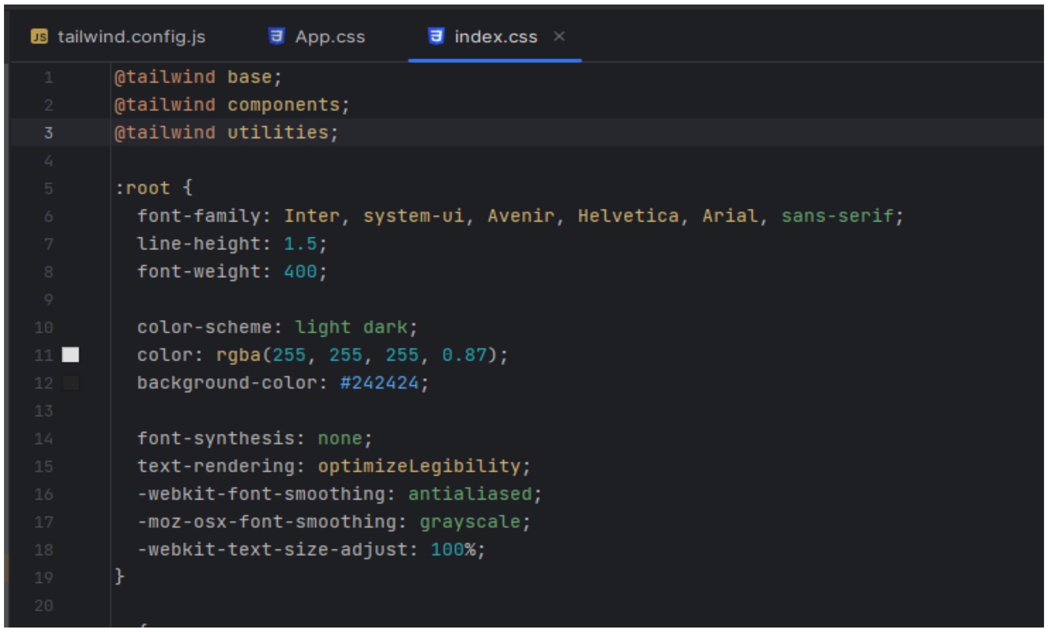
```
Done in 4.8s
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject\k5_frontend> pnpx tailwindcss init -p
Packages: +81
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Progress: resolved 82, reused 81, downloaded 0, added 81, done

Created Tailwind CSS config file: tailwind.config.js
Created PostCSS config file: postcss.config.js
(djangoproject-py3.11) PS C:\Users\Svene\.dev\projects\djangoProject\k5_frontend>
```

After this, to configure the tailwind for the project, the configuration can add the paths to all of the template files in the **tailwind.config.js** file.



```
/** @type {import('tailwindcss').Config} */
no usages
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Adding Directives for Tailwind. To include Tailwind's base, components, utilities, and variants styles into the CSS file located at "./src/index.css," it is necessary to add the appropriate @tailwind directives for each of Tailwind's layers. These directives serve as a reference for the custom functions and directives that Tailwind provides for your CSS. These directives are Tailwind-specific at-rules designed to offer special functionality for Tailwind CSS projects, as outlined in the Tailwind documentation.



```
@tailwind base;
@tailwind components;
@tailwind utilities;

:root {
  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;
  line-height: 1.5;
  font-weight: 400;

  color-scheme: light dark;
  color: rgba(255, 255, 255, 0.87);
  background-color: #242424;

  font-synthesis: none;
  text-rendering: optimizeLegibility;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  -webkit-text-size-adjust: 100%;
}

```

ViteJS Integration Problem. Since I am using ViteJS, the build command to public instead of dist (in ViteJS) will encounter the problem:

```
[28/Sep/2023 21:59:48] "GET / HTTP/1.1" 200 452
Not Found: /assets/index-8b5d1e09.css
[28/Sep/2023 21:59:48] "GET /assets/index-8b5d1e09.css HTTP/1.1" 404 2270
Not Found: /assets/index-5ade48c9.js
[28/Sep/2023 21:59:48] "GET /assets/index-5ade48c9.js HTTP/1.1" 404 2267
Not Found: /vite.svg
[28/Sep/2023 21:59:49] "GET /vite.svg HTTP/1.1" 404 2219
```

Fortunately, there was a way that was discovered of addressing the problem, which is to update the vite.config.js file and add the following code:

```
export default defineConfig({
  build: {
      outDir: './public',
  },
});
```

Django and ReactJS Integration. After finishing setting up the frontend and backend scaffolding and configuration, there is a need to integrate both the frontend and backend. This can be done by adding the following code into the settings.py in the Django project file.

```
import os
...
'DIRS': [BASE_DIR / 'templates', os.path.join(BASE_DIR, 'k5_frontend/dist')]
...
STATICFILES_DIRS = [ os.path.join(BASE_DIR, 'k5_frontend/public') ]
```

Then create a view.py in the main app, and it must have the following:

```
from django.shortcuts import render
def index(request):
    return render(request, 'index.html')
```

After that, the urls.py must be modified:

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
        path('admin/', admin.site.urls),
        path('', views.index, name='index')
]
```

Django and MongoDB Integration. This is a quick overview of how the POC of the Django and MongoDB integration was implemented. A new project and cluster in MongoDB Atlas was created. Then, using MongoDB Compass, it related to the Atlas Cluster and Setup the Connection Security: un: aidrecabrera, pw: [Redacted]. Not all the steps f the procedure will be covered due to the length of the procedure.

```
import os
...
'DIRS': [BASE_DIR / 'templates', os.path.join(BASE_DIR, 'k5_frontend/dist')]
...
STATICFILES_DIRS = [ os.path.join(BASE_DIR, 'k5_frontend/public') ]
```

Install driver by running the following on the command line

```
    $ python -m pip install pymongo
```

Add connection string into application code

```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi
uri =
"mongodb+srv://aidrecabrera:<password>@atlascluster.mdu4pqz.mongodb.net/?retryWrites=true
&w=majority"
# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi('1'))
# Send a ping to confirm a successful connection
try:
    client.admin.command('ping')
    print("Pinged your deployment. You successfully connected to MongoDB!")
except Exception as e:
    print(e)
```

REFERENCES

[1] United Nations, “Philippines: Indigenous knowledge takes on climate crisis | UN News.” Accessed: Oct. 03, 2023. [Online]. Available: https://news.un.org/en/story/2023/09/1140337

[2] United Nations, “The Climate Crisis – A Race We Can Win,” United Nations. Accessed: Oct. 03, 2023. [Online]. Available: https://www.un.org/en/un75/climate-crisis-race-we-can-win

[3] R. J. Nicholls *et al.*, “Sea-level rise and its impacts on coastal systems,” *Nature*, vol. 502, no. 7474, pp. 575–587, 2011.

[4] “World Bank Climate Change Knowledge Portal.” Accessed: Oct. 03, 2023. [Online]. Available: https://climateknowledgeportal.worldbank.org/

[5] “Philippines country most at risk from climate crisis.” Accessed: Oct. 03, 2023. [Online]. Available: https://www.amnesty.org.uk/philippines-country-most-risk-climate-crisis

[6] World Resources Institute, “The Philippines Climate Watch 2022: Country Profile,” 2022.

[7] Department of Environment and Natural Resources, “Philippine Climate Change Adaptation Plan,” 2022.

[8] “The Paris Agreement | UNFCCC.” Accessed: Oct. 03, 2023. [Online]. Available: https://unfccc.int/process-and-meetings/the-paris-agreement

[9] L. Peeples, “Lessons from the COVID data wizards,” *Nature*, vol. 603, no. 7902, pp. 564–567, Mar. 2022, doi: 10.1038/d41586-022-00792-2.

[10] J. Perkel, “Behind the Johns Hopkins University coronavirus dashboard,” Nature Index. Accessed: Oct. 03, 2023. [Online]. Available: https://www.nature.com/nature-index/news/behind-the-johns-hopkins-university-coronavirus-dashboard

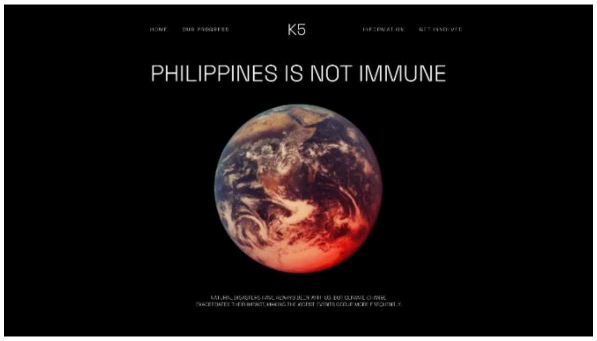
[11] E. Mathieu *et al.*, “Coronavirus Pandemic (COVID-19),” *Our World Data*, 2020.

[12] “Software Engineering | Iterative Waterfall Model,” GeeksforGeeks. Accessed: Oct. 03, 2023. [Online]. Available: https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/

[13] V. Chandra, “Comparison between various software development methodologies,” *Int. J. Comput. Appl.*, vol. 131, no. 9, pp. 7–10, 2015.

[14] S. McLeod, “Interrelated Attributes of Project Feasibility: Visualizing the $^{\text{TEL}}$OS Framework,” *Sci. Posters*, 2021.

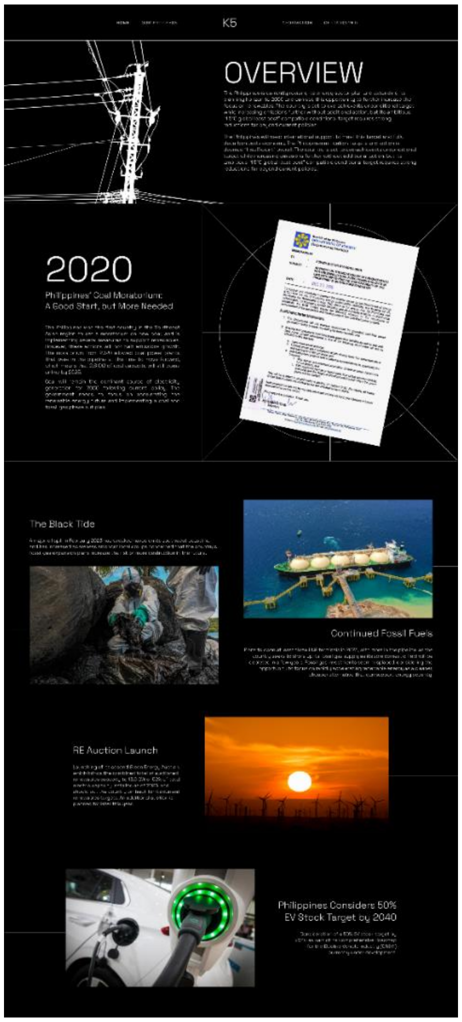
APPENDIX I. TENTATIVE DESIGN



HOME OUR PROGRESS K5 INFORMATION GET INVOLVED

PHILIPPINES IS NOT IMMUNE

NATURAL DISASTERS HAVE ALWAYS BEEN WITH US BUT CLIMATE CHANGE EXACERBATES THEIR IMPACT, MAKING THE WORST EVENTS OCCUR MORE FREQUENTLY.



HOME OUR PROGRESS K5 INFORMATION GET INVOLVED

OVERVIEW

[illegible]

[illegible]

2020

Philippines' Coal Moratorium: A Good Start, but More Needed

[illegible]

[illegible]

The Black Tide

[illegible]

Continued Fossil Fuels

[illegible]

RE Auction Launch

[illegible]

Philippines Considers 50% EV Stock Target by 2040

[illegible]

Provisional Sample of the COVID Pulse Home Page (1920 x 3224)