Canvas to Google

Assignment Calendar Connections

Farley, Gallagher, Utter

Project Utilized:

- Python
- TKinter
- SQLAlchemy
- Pydantic
- Google Cloud APIs
- JSON
- FastAPI

Getting Classes and Assignments

- Utilize a request statement to access the Canvas website
- auth_header is the Canvas Token

```
# Iterate over the filtered courses
for course in specific_course_ids:
    # Make a request to get assignments for the current course
    assignments_url = BASE_URL + f'/api/v1/courses/{specific_course_ids[count]}/assignments'
    assignments_params = {"per_page": str(PER_PAGE)}
    assignments_request = requests.get(assignments_url, headers=auth_header, params=assignments_params)
    assignments_request.raise_for_status()

    # Add assignments to the list
    assignments = assignments_request.json()
```

Storing in a DataFrame

- Assignments then get placed into a DataFrame to make creating Google Calendar events easy

```
assignments_df = pd.DataFrame(assignments)
assignments_result = assignments_df.to_string(
    columns=['id', 'name', 'due_at', 'points_possible']
)
print(f"Assignments for {specific_course_ids[count]}:")
print("----------------------------")
print(assignments_result)
print("\n")
```

Inserting Assignments

- Grab name and time from Canvas, then create an event to put into the calendar

```python
# Create an event in Google Calendar
event = {
    'summary': assignment_name,
    'description': f'Assignment due: {assignment_name}',
    'start': {'dateTime': due_date.isoformat(), 'timeZone': 'UTC'},
    'end': {'dateTime': (due_date + timedelta(hours=1)).isoformat(), 'timeZone': 'UTC'},
}

# Insert the event
event = service.events().insert(calendarId='primary', body=event).execute()

print(f'Event created: {event.get("htmlLink")}')
```

Hosted Server for the backend

- Code that starts a server for the database

```python
conn_string = f"postgresql://{USERNAME}:{PASSWORD}@{HOST}/{NAME}"
engine = create_engine(conn_string)
BaseTable.metadata.create_all(bind=engine)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engin

router = APIRouter()
```

5 usages JackGallagher41

```python
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

```python
from fastapi import FastAPI

from api import userApis, courseApis, assignmentApis

app=FastAPI()

app.include_router(userApis.router, prefix="/users", tags=["Users"])
app.include_router(courseApis.router, prefix="/courses", tags=["Courses"])
app.include_router(assignmentApis.router, prefix="/assignments", tags=["Assignments"])
```

Tables created in Python using SQLalchemy

```python
class User(BaseTable):
    __tablename__ = "users"
    uid = Column(Integer, primary_key=True, index=True)
    username = Column(String)
    canvasAccessToken = Column(String)
    googleCalendarAccessToken = Column(String)
```

```python
class Assignment(BaseTable):
    __tablename__="assignments"
    assignmentID = Column (Integer, primary_key=True, index=True)
    title= Column(String)
    description= Column(String)
    dueDate = Column(DateTime)
    courseID=Column(Integer)
```

JSON Models

- Example of models used in correlation with rest api to put the data into a JSON format

```python
class userCreate(BaseModel):
    username: str
    canvasAccessToken: str
    googleCalendarAccessToken: str
```

4 usages JackGallagher41

```python
class courseResponse(BaseModel):
    courseID: int
    courseName: str
    profFName: str
    profLName: str
    crn: str
    uid: int
```

Rest APIS

- Example of rest apis used to communicate the ui form and canvascomms with the database

```python
@router.delete( path: "/{user_id}", response_model=str)
async def deleteUser(user_id: int, session: Session = Depends(get_db)):
    userToDelete = session.query(User).filter(User.uid == user_id).first()
    if userToDelete:
        session.delete(userToDelete)
        session.commit()
        return f"User with the following ID has been deleted: {user_id} "
    else:
        raise HTTPException(status_code=404, detail="User not found")
```

```python
# API that gets a user based off their user id
JackGallagher41
@router.get( path: "/id_retrieve/{user_id}", response_model=userResponse)
async def getUser(user_id: int, session: Session = Depends(get_db)):
    user = session.query(User).filter(User.uid == user_id).first()
    if user is None:
        raise HTTPException(status_code=404, detail="User not found")
    return user
```

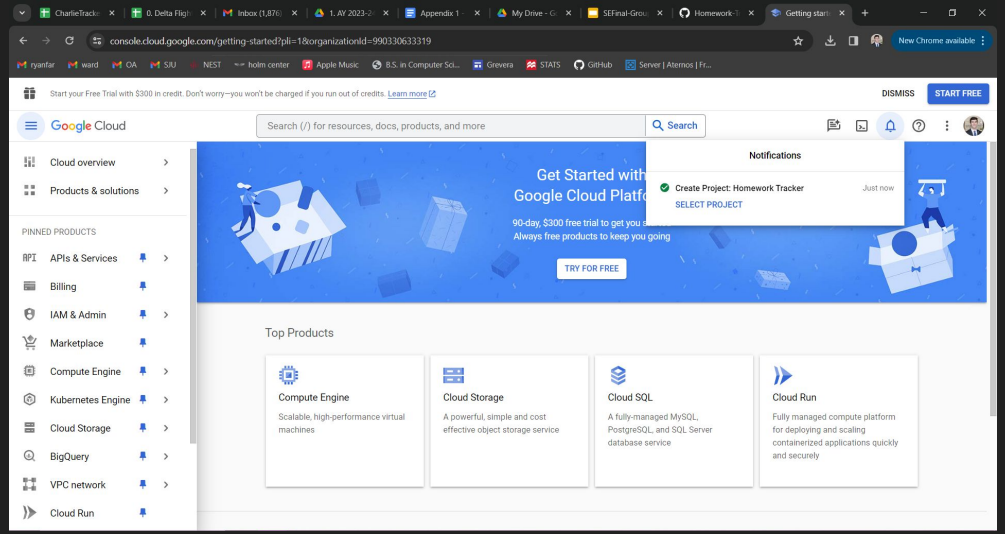
```python
# API that gets a user based off their Canvas Token
JackGallagher41
@router.get( path: "/CAT_retrieve/{user_CAT}", response_model=dict)
async def getUser(user_CAT: str, session: Session = Depends(get_db)):
    user = session.query(User).filter(User.canvasAccessToken == user_CAT).first()
    if user is None:
        raise HTTPException(status_code=404, detail="User not found")
    uid= {"user_uid": user.uid}

    return uid
```

```python
@router.post( path: "/create/{create_user}", response_model=userResponse)
async def createUser(user: userCreate, session: Session = Depends(get_db)):
    newUser = User(**user.dict())
    session.add(newUser)
    session.commit()
    session.refresh(newUser)
    return newUser
```

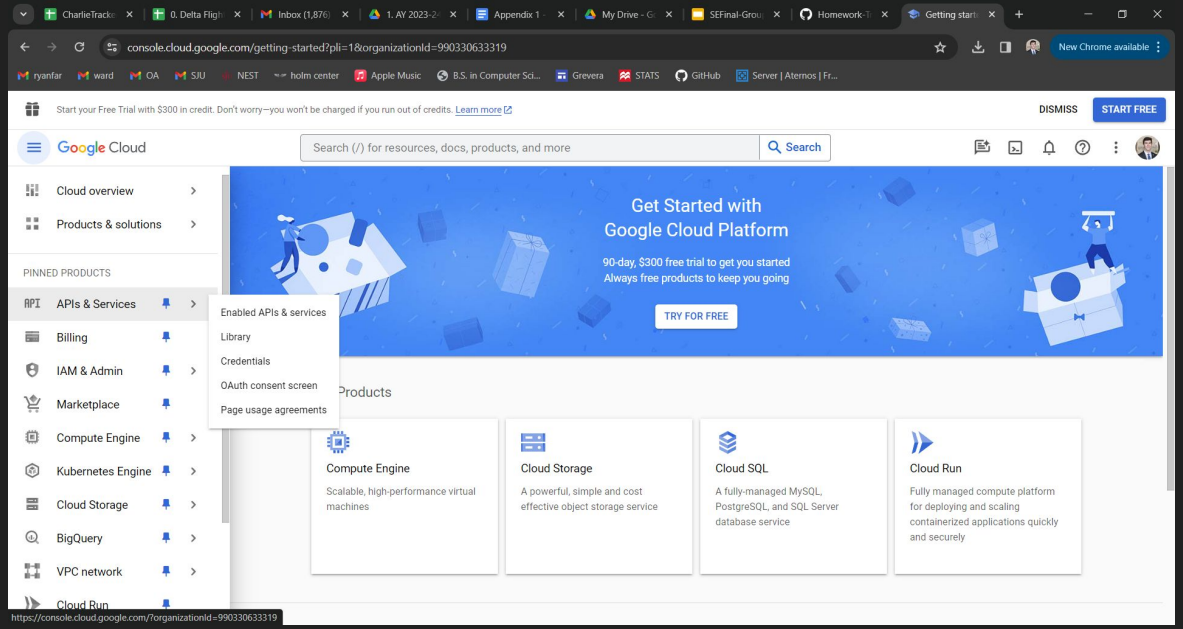
Google API Steps

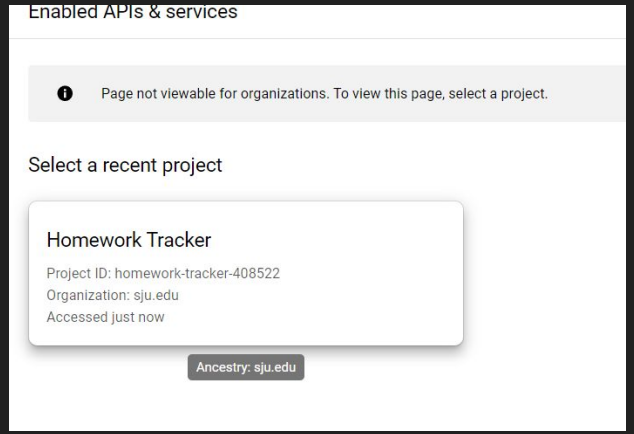
Steps for getting your Google Path JSON

1. Search in your browser 'Google Cloud Console' and click on Google's link
2. Click on the 'Select a Project' button and change the organization at the top to sju.edu, then click 'NEW PROJECT'
3. Name the project 'Homework Tracker' and leave the location and organization as sju.edu, then create

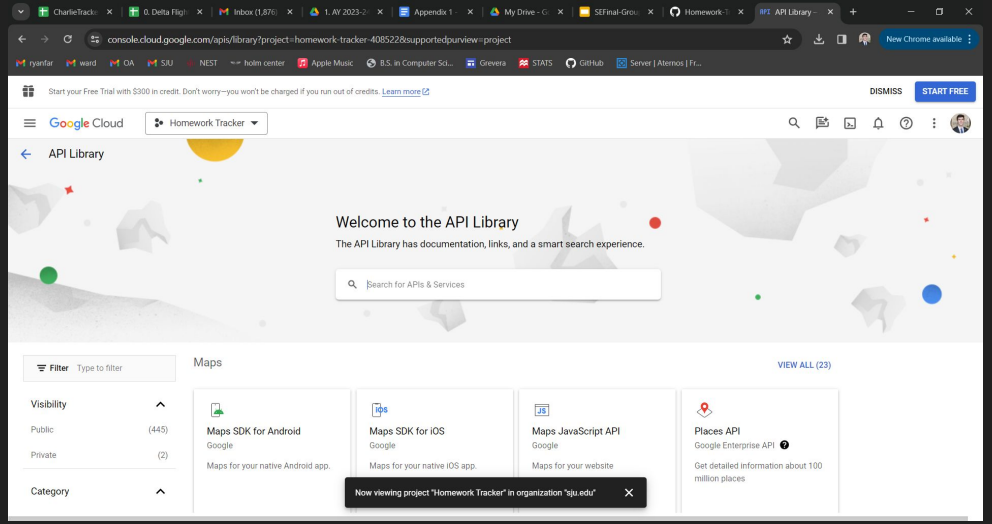


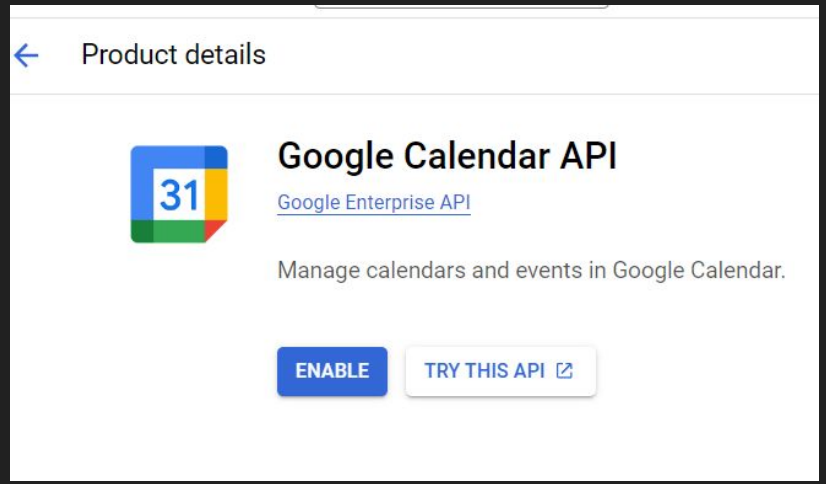
4. Now navigate to the left side of your screen to 'API's and Services', then select your project. At the top hit '+ ENABLE APIS AND SERVICES'

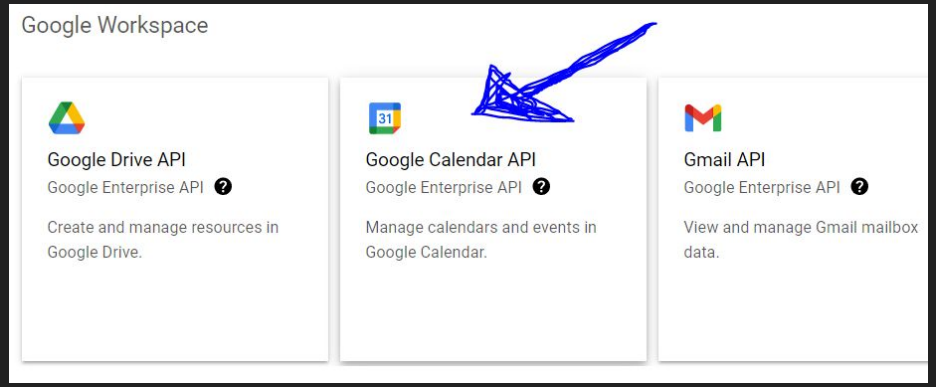




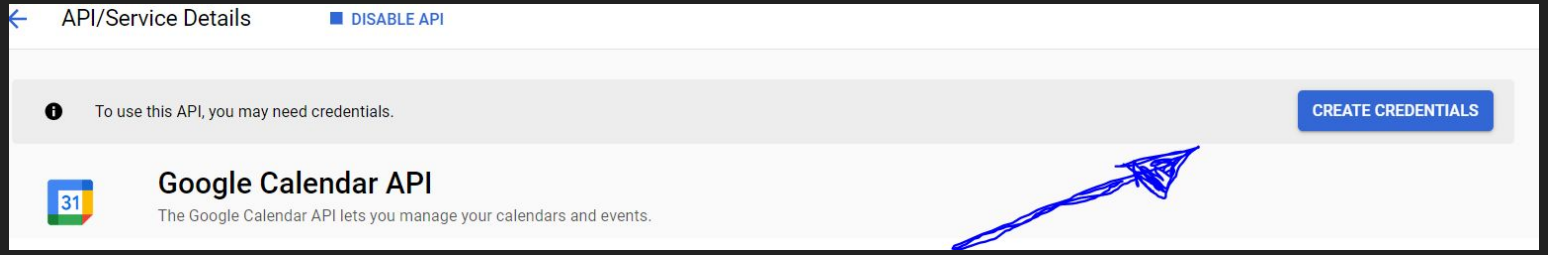
5. Scroll down to the Google Calendar API, click on it and hit 'ENABLE'

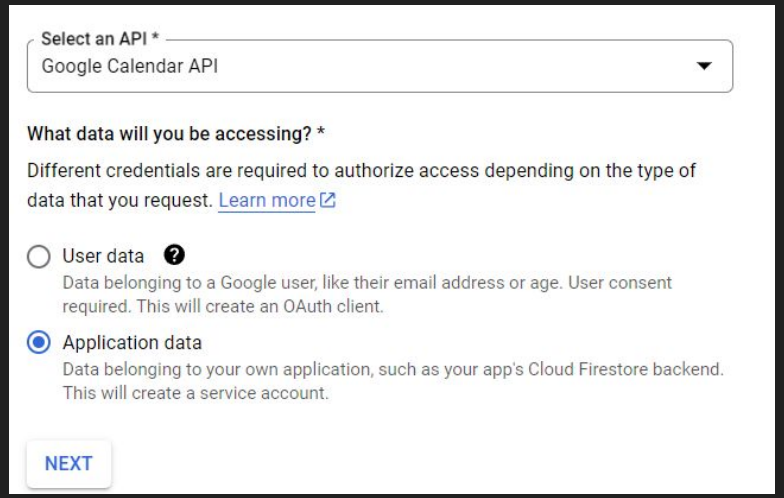




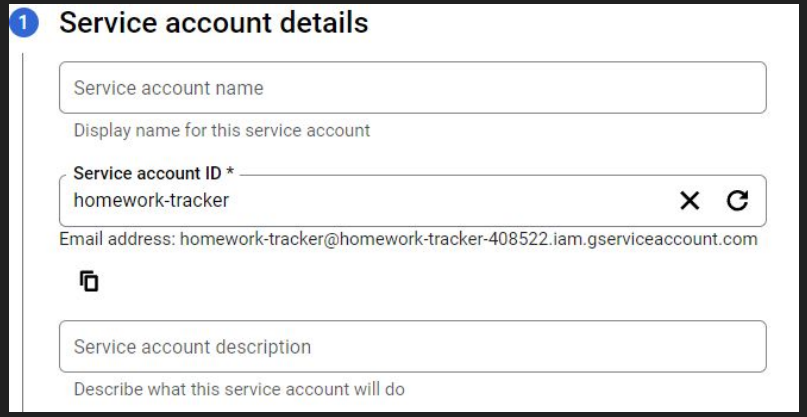


6. Now hit 'CREATE CREDENTIALS' on the top of your screen. Leave the API on Google Calendar and select 'Application data' and hit 'NEXT'

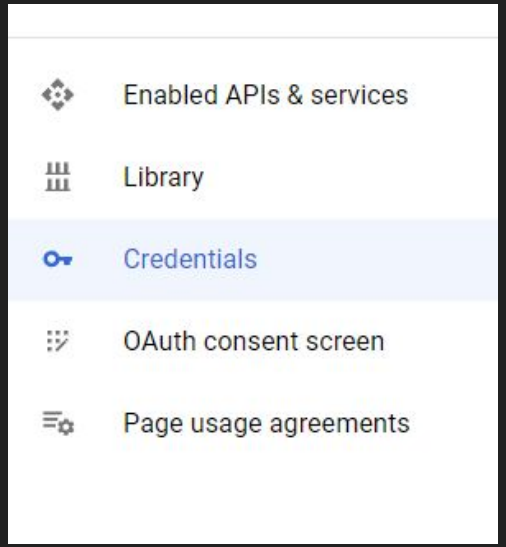




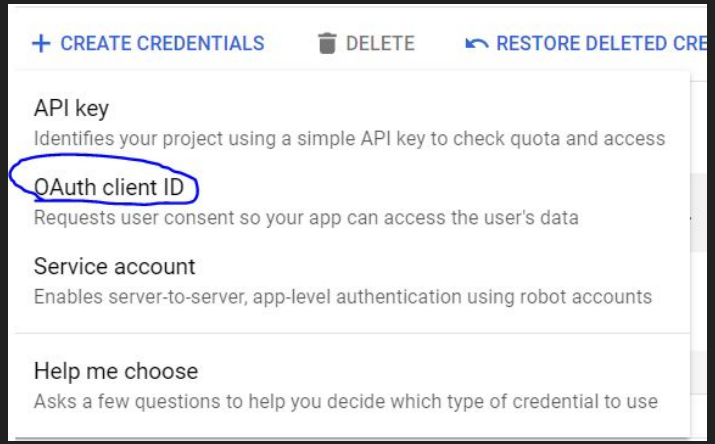
7. Name it the same as what you named the project then skip past the optional parts and finalize it

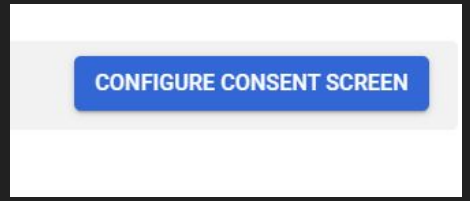


8. Now navigate to the 'Credentials' tab on the left side of your screen and create another credential by hitting the '+ CREATE CREDENTIALS' at the top of your screen

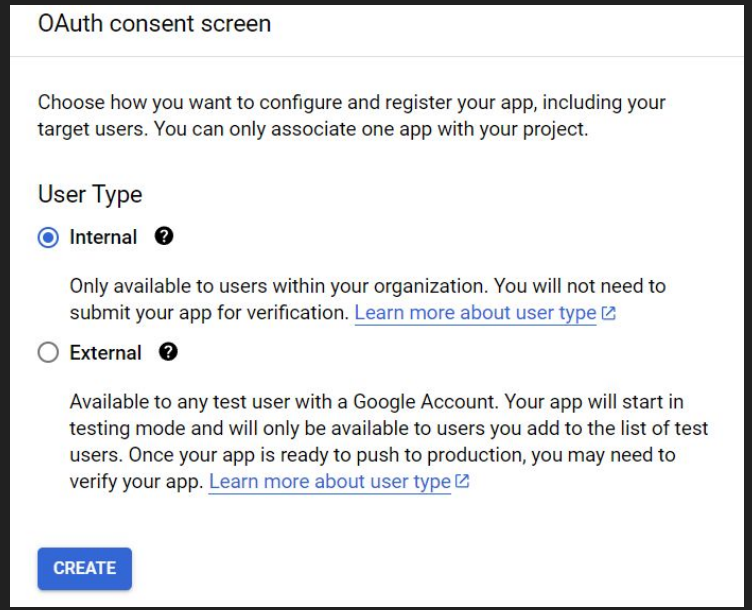


9. Click 'OAuth Client ID' and then hit 'Configure Consent Screen'





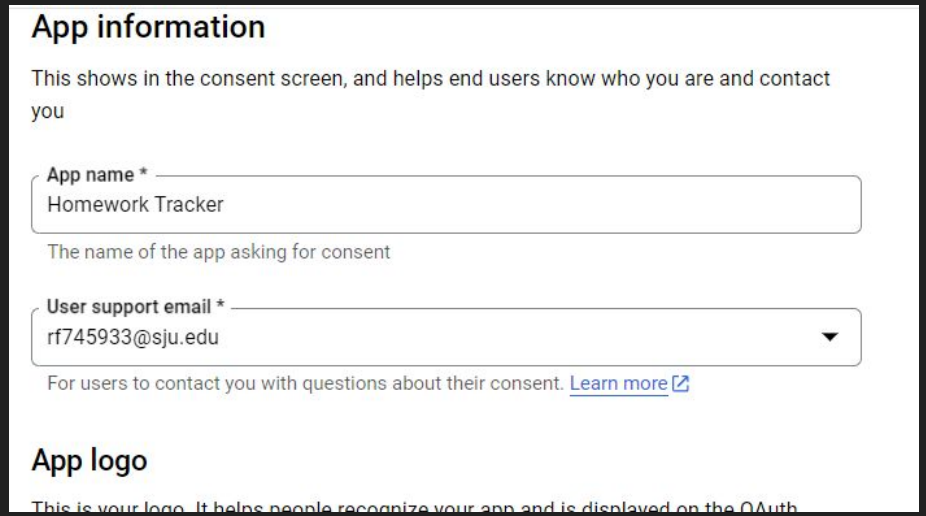
10. Make sure you click 'Internal' then create.



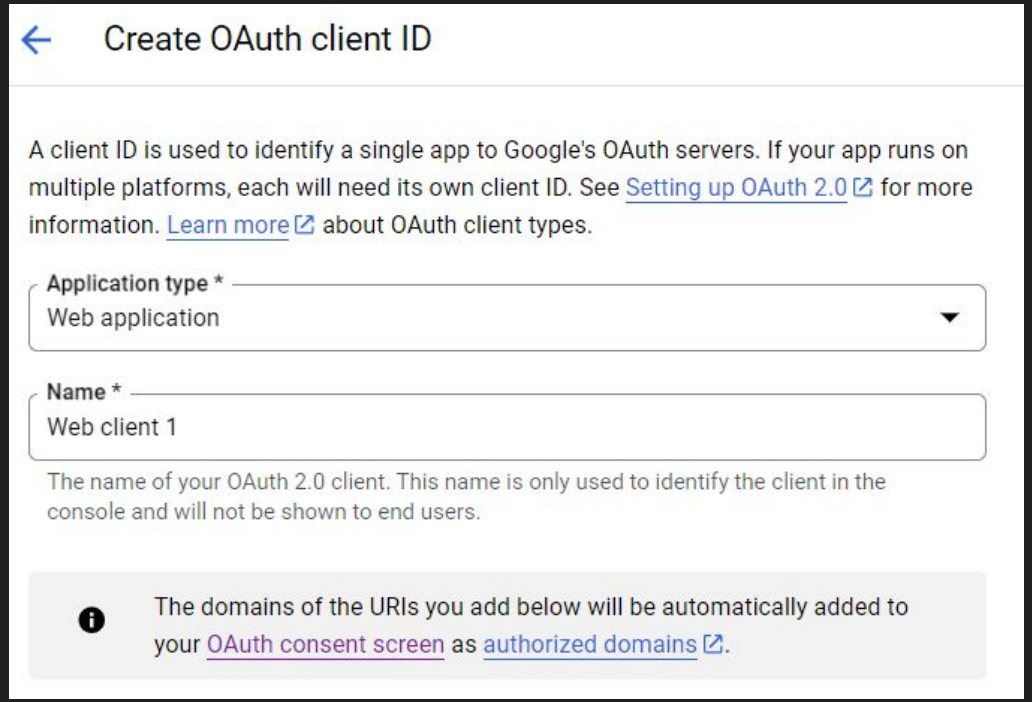
11. Type in the same project name you have been using into the 'App name' box and then use your school email in the following two mandatory boxes

12. Skip past the 'Scopes' page and finalize the consent screen.

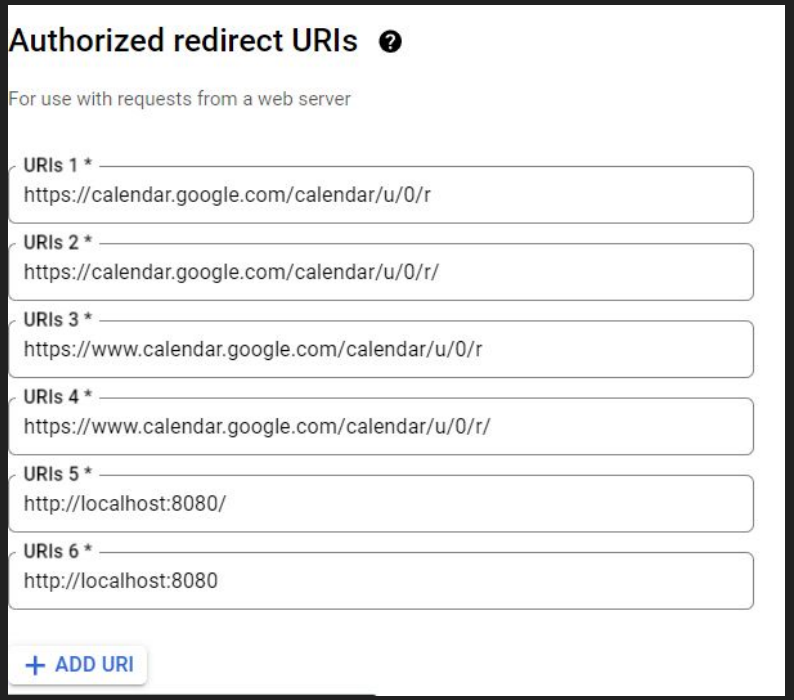
13. Repeat step #8

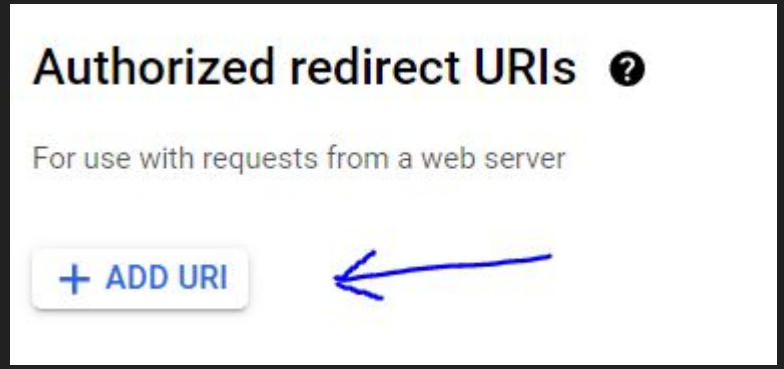


14. Click 'OAuth Client ID' again and put the application type to Web Application and name it



15. Scroll down until you see Authorized URI's. You are going to want to make six. Make sure your six are:





16. Once it's created you should get a pop-up confirming creation. At the bottom of that pop-up hit 'DOWNLOAD JSON'

17. Find that downloaded file in your downloads and copy the path of the file.