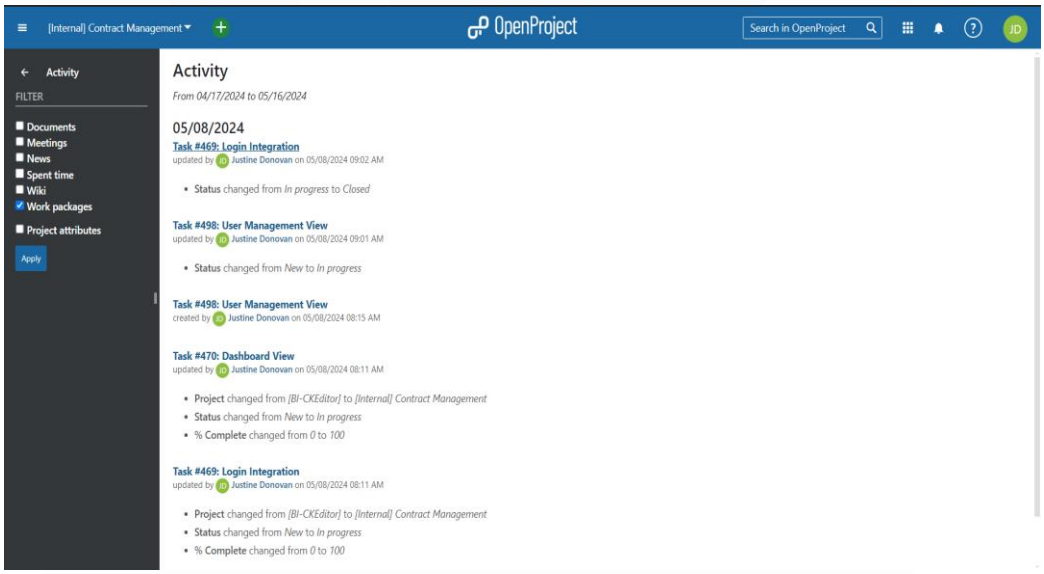
BAB III
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pada Infracom Technology terdapat *Learning Capability Development* yang dibgai menjadi beberapa tim guna untuk membagi pekerjaan dari masing-masing divisinya. *Learning Capability Development* terbagi menjadi beberapa tim yaitu, *Network Engineering*, *Appplication Developer, Cloud Native Database*, *Data Analytics* dan *Solution Architech*. Mahasiswa ditempatkan pada divisi *Application Developer*. Tugas utama *Application Developer* yaitu membuat aplikasi baru untuk *client* dan meningkatkan kemampuan *softskill* individu sehingga mampu meningkatkan kualitas aplikasi yang dibuat. Dalam divisi *Application Developer* biasanya terbagi menjadi 2 bagian, yaitu *Backend Developer* dan juga *Frontend Developer* dan terdapat 1 *Team Lead* yang memiliki tugas untuk meningkatkan kualitas aplikasi, memberikan ilmu terhadap anggota mengenai teknologi / *tools* yang digunakan dan memberikan informasi-informasi yang penting dalam pengembangan aplikasi terutama pada tahap *development*. Alur koordinasi kerja yang berjalan pada divisi *Application Developer* yaitu dengan menambahkan *activity* baru pada *website* Open Project. Open Project merupakan sebuah *web application* yang berguna untuk melakukan *track* terhadap proyek yang sedang berjalan. Dengan melakukan input *task* baru dalam Open Project maka akan lebih mudah diketahui masing masing tugas yang sedang dikerjakan dari anggota tim. Untuk pembagian tugas pada saat proyek berjalan *team lead* akan membagikan tugas kepada anggota sehingga setiap anggota mendapat porsi yang sama saat proyek berjalan, mahasiswa lebih banyak menangani masalah dalam *service* aplikasi atau membuat bagian *backend* dalam pembuatan *backend* mahasiswa berkomunikasi dengan *frontend* untuk menentukan seperti apa *response* maupun *request* yang dibutuhkan dalam aplikasi nantinya. Lalu setelah itu *team lead* akan membantu anggota jika mengalami kesulitan dan memberikan saran mengenai program yang dibuat oleh anggota.

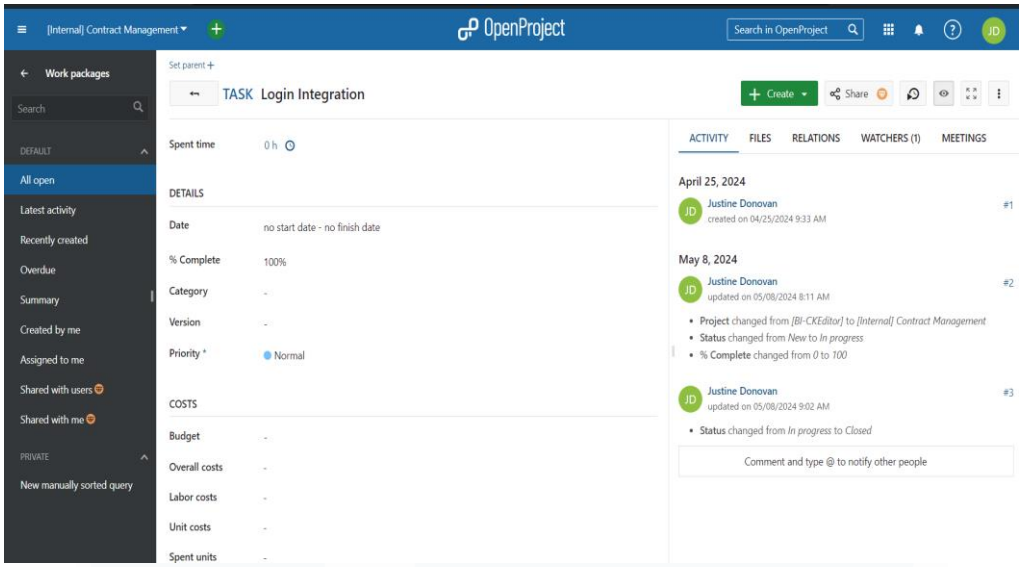
Gambar 3. 1 Tampilan Login Open Project

Task/*activity* yang ditambahakan juga dapat membantu mahasiswa dalam melihat fitur apa yang sedang dikerjakan atau dalam tahap *develop*.

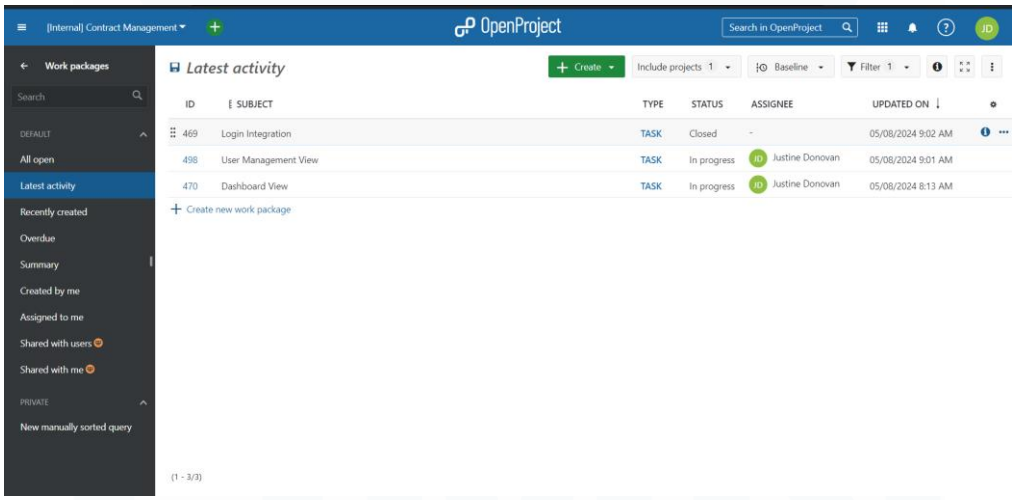


Gambar 3. 2 Tampilan List Project

Sesudah melakukan input *Task* baru mahasiswa akan mengerjakan sesuai dengan *task* yang diinput. Saat *task* sedang dalam masa *develop* mahasiswa akan melakukan update secara berkala dalam *website Open Project*.

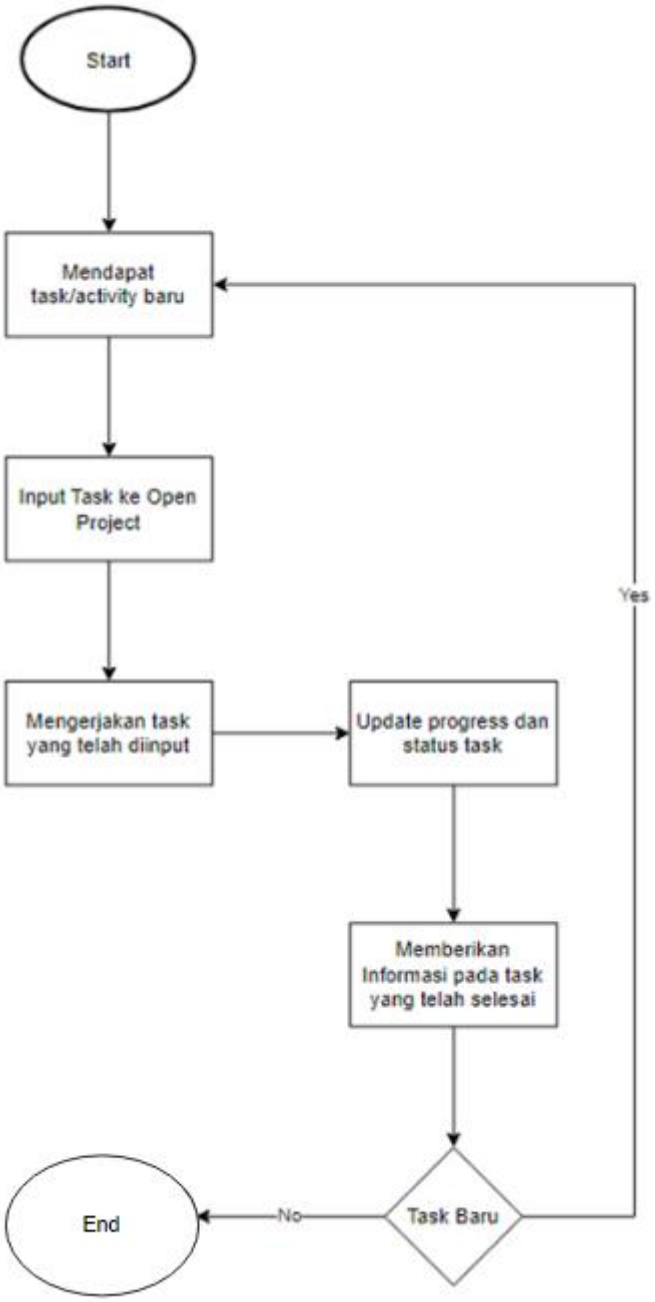


Gambar 3. 3 Tampilan Log masing masing *task/activity*



Gambar 3. 4 Tampilan List Task Open Project

Setelah melakukan dan tahap pada open project telah selesai maka mahasiswa memberi informasi kepada *team lead* dan akan diberikan *task/activity* baru. Lalu untuk waktu *meeting internal* diadakan setiap hari selasa, yang bertujuan untuk membahas hasil / informasi mengenai sebuah proyek dan juga guna untuk melakukan *report* kepada atasan. Jika dengan flowchart, maka berikut hasil dari alur yang bekerja.



Gambar 3. 5 Alur Koordinasi

Pada Gambar 3.5 menunjukan alur koordinasi yang berjalan pada tim *application developer* menggunakan *platform* tambahan yaitu *Open Project* pada alur diawali dengan melihat *task* baru yang diterima oleh anggota lalu dilanjutkan dengan melakukan input secara manual pada platform *Open Project* yang selanjutnya

dengan pengerjaan *task* tersebut sehingga dapat melakukan *update* terhadap *progress* yang sedang berjalan sesuai dengan *task* yang diinput. Setelah *task* telah selesai dikerjakann selanjutnya dengan mengganti status yang berada pada bagian *task detail*.

3.2 Tugas dan Uraian Kerja Magang

Selama periode magang mahasiswa berkontribusi dalam mengerjakan 2 proyek dan satu kewajiban untuk mengambil serifikat. Proyek proyek akan dibgai menjadi 2 kategori yaitu proyek internal dan external. Proyek internal bertujuan untuk membuat aplikasi yang digunakan dalam kantor dan proyek external bertujuan untuk membuat aplikasi langsung kepada *client* berupa sebuah instansi atau perusahaan lain. Untuk proyek-proyek yang berjalan mayoritas berbasis web teknologi yang digunakan relatif lebih banyak dan luas. Tabel dibawah ini merupakan rincian kerja magang yang dilakukan oleh mahasiswa

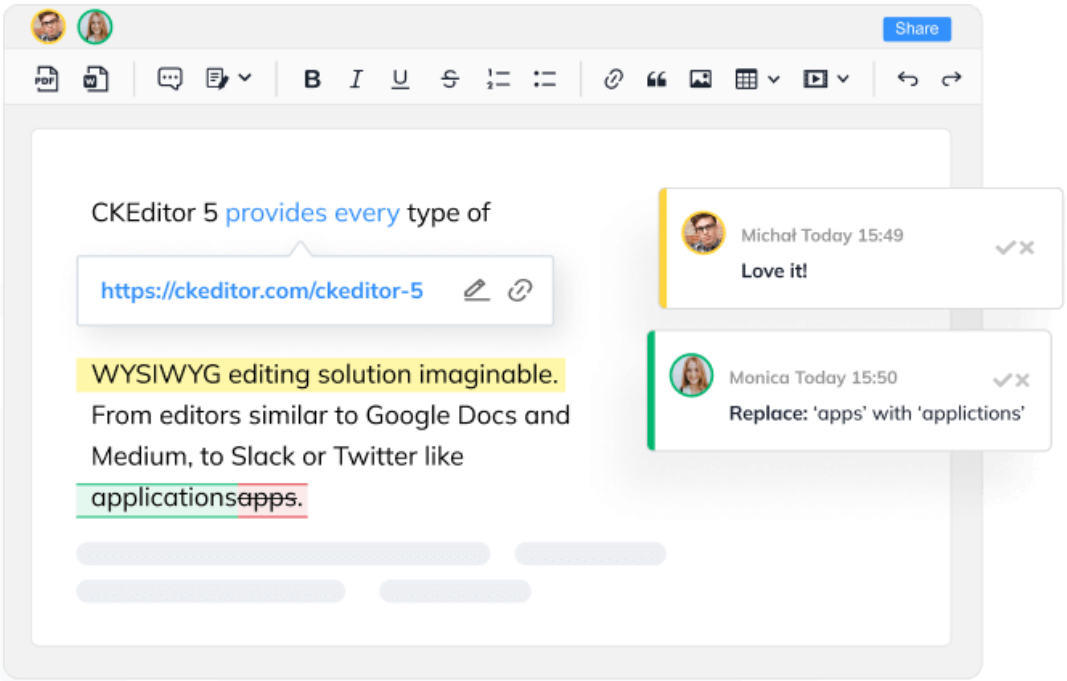
Tabel 3. 1 Perincian Tugas Kerja Magang

No	Kegiatan	Mulai	Selesai	Minggu
1	Training Mendix	2024-01-08	2024-03-05	1-9
1.1	Mengerjakan *Course Academy* Mendix	2024-01-08	2024-03-05	1-9
1.2	Pengambilan sertifikasi Mendix	2024-02-02	2024-02-05	5
2	Proyek CKEditor	2024-02-12	2024-03-21	6-11
2.1	Membuat API Auth	2024-02-12	2024-02-15	6
2.2	Membuat API *Cloud Services*	2024-02-16	2024-02-28	7-8
2.3	Membuat API CKBOX	2024-02-29	2024-03-12	9-10
2.2	Melakukan Testing API	2024-03-12	2024-03-21	10-11
3	Proyek Contract Management	2024-03-22	2024-05-29	12-20

3.1	Setup database dan environment	2024-03-22	2024-03-25	12
3.2	Membuat API Proyek	2024-03-25	2024-04-08	12-13
3.3	Melakukan Testing API Proyek	2024-04-08	2024-04-12	13
3.4	Melakukan konfigurasi ulang database	2024-04-15	2024-04-15	14
3.5	Membuat API Product dan Master Data	2024-04-16	2024-04-19	14
3.6	Testing API Product dan Master Data	2024-04-19	2024-04-25	14-15
3.7	Integrasi Pocketbase dengan LocalDB	2024-04-26	2024-04-30	15
3.8	Melakukan *FETCH* APi di bagian Frontend	2024-04-30	2024-05-08	16-17
3.9	Membuat Dokumentasi	2024-05-06	2024-05-17	17-18
3.10	Melakukan Testing aplikasi dan Fixed Bug	2024-05-20	2024-05-29	18-20

Lalu untuk lebih detailnya berikut merupakan kontribusi dalam masing masing proyek dan juga pelatihan Mendix yang berjalan pada saat magang:

1. **CKEditor** – Web Application
   CKEditor merupakan sebuah *rich text editor* yang dimana ini merupakan sebuah library yang dapat digunakan guna untuk membuat sebuah *text editor* seperti Google Docs dan Ms.Word. CKEditor *editor* WYSIWYG (*What You See is What You Get*) [6].



Gambar 3. 6 Contoh tampilan CKEditor

Dalam proyek ini mahasiswa berkontribusi dalam membuat API endpoint berdasarkan *service* API CKEditor. Pada proyek ini *tools* yang digunakan untuk membuat aplikasi *backend* untuk *framework* yang digunakan adalah FastAPI, dengan menggunakan FastAPi dapat mempercepat proses development aplikasi sehingga juga aplikasi yang dibuat dapat memiliki dokumentasi yang lengkap dan fitur yang modern [7].

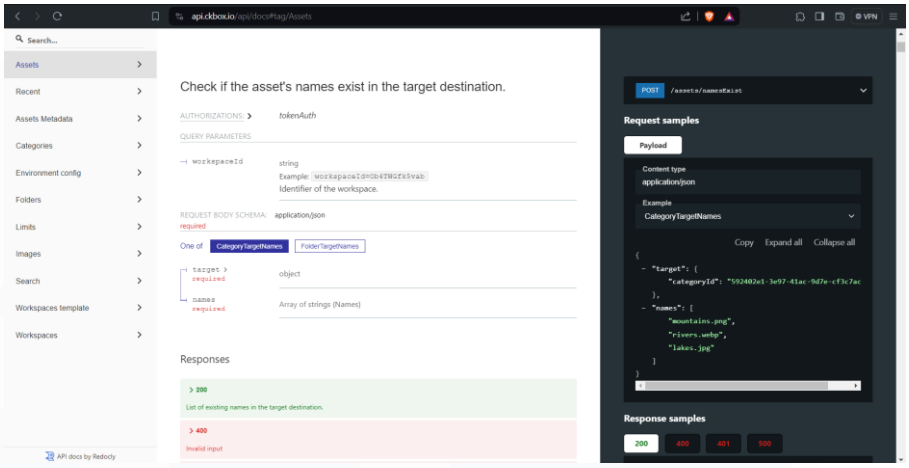
Gambar 3. 7 Logo FastAPI

FastAPI sendiri digunakan dalam bahasa Python, sehingga sintaks sintaks ataupun *function* yang digunakan dapat dipahami dengan baik [7]. Lalu dengan adanya 2 service yang berbeda antara *service* CKEditor dan *service* yang dibuat sebagai penghubung antara database dengan service CKEditor. Dalam melakukan integrasi terhadap aplikasi yang dibangun dengan service CKEditor, harus menggunakan *CORS Middleware*, *CORS* sendiri merupakan *cross origin resource sharing* guna untuk bertukar data antar resource, yaitu CKEditor dengan aplikasi *backend* yang dibuat. Berikut merupakan contoh penggunaan *CORS*.

```python
from fastapi.middleware.cors import CORSMiddleware
ALLOWED_ORIGINS = ["*"]

app.add_middleware(
    CORSMiddleware,
    allow_origins=ALLOWED_ORIGINS,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"]
)
```

Gambar 3. 8 Contoh penggunaan CORS pada FastAPI

Pada Gambar 3.8 merupakan kode untuk melakukan konfigurasi terhadap CORS yang dapat mengakses *service* yang dibuat, mahasiswa membuat service dapat diakses oleh segala akses demi kelancaran dalam masa *develop* saat tahap develop telah selesai maka CORS akan diset sesuai dengan domain yang di hosting. Untuk mahasiswa dapat mengetahu seperti apa *request* dan *response* yang dibutuhkan oleh *service* CKEditor yaitu dengan membuka dokumentasi yang disediakan oleh CKEditor.



Gambar 3. 9 Dokumentasi API CKEditor

CKEditor memiliki beberapa API yang dibagi menjadi beberapa bagian atau dibangun secara *Microservices*, berikut merupakan *service* yang disediakan oleh CKEditor:

1. **CKBox Restfull API**

   CKBox merupakan penyimpanan yang disediakan oleh CKEditor, dengan menggunakan API CKBox maka *developer* dapat menyimpan *file* ataupun *assets* yang digunakan dalam sebuah *text editor* sehingga dapat digunakan secara berulang tanpa harus memakan penyimpanan yang lebih besar. Keperluan *header request* yang dibutuhkan oleh setiap *service*nya memiliki perbedaan tersendiri CKBox memerlukan HTTP: tokenAuth sebagai *header* yang digunakan untuk melakukan *request* data kepada *service* CKBox [6]. *Header* yang dibutuhkan yaitu "Authorization" yang berisi JWT(*JSON Web Token*) yang digunakan sebagai *identifier* oleh CKEditor. Ada 3 buah tipe hasil *headers* yang diparsing [8].

   - **Regular User**

```
{
 "aud": "xPpltkveX59E8QVKdNMg",
```

```
  "sub": "f2281f2b-dae9-45b4-9941-e0be16af24b0"
}
```

` Untuk *regular user* hanya membutuhkan 2 properti sebagai *identifier* yaitu *aud* dan *sub*

- **Admin User**

  Sedangkan untuk *admin user* yang digunakan akan menambah 1 properti dan subproperti yang lainnya sebagai *identifier* oleh CKBox *services*

```
{
  "aud": "xPpltkveX59E8QVKdNMg",
  "sub": "f2281f2b-dae9-45b4-9941-e0be16af24b0",
  "auth": {
      "ckbox": {
        "role": "admin",
        "workspaces": [ "pHUSQFj_QIvcfNMCB9Pp" ]
      }
  }
}
```

Gambar 3. 10 Contoh *request* admin

- **Superadmin User**

  Sedangkan untuk *superadmin user* yang digunakan akan menambah 1 properti dan 1 subproperti sebagai *identifier* oleh CKBox *services*

```
{
  "aud": "xPpltkveX59E8QVKdNMg",
  "sub": "f2281f2b-dae9-45b4-9941-e0be16af24b0",
  "auth": {
      "ckbox": { "role": "superadmin" }
  }
}
```

Gambar 3. 11 Contoh *request* super admin

Dalam CKBox *Restfull* API terdapat berbagai macam endpoint yang dapat digunakan oleh *developer* dalam menghubungkan CKBox dengan

program yang akan dibuat. Berikut merupakan *list* dari *endpoint* yang disediakan oleh CKBox [8]:

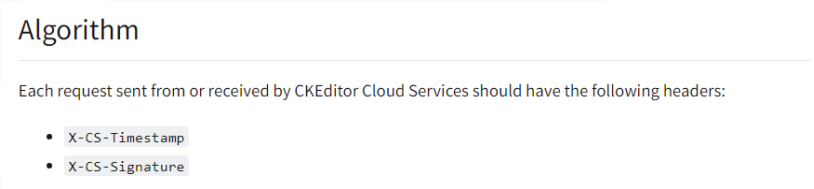
- Assets
- Recent
- Assets Metadata
- Categories
- Environtment Config
- Folders
- Limits
- Images
- Search
- Workspaces template
- Workspace

API dapat dilihat dalam website dokumentasi resmi CKBox dari CKEditor [8].

2. CKEditor Cloud Services Restful APIs

Cloud Service merupakan salah satu service yang ditawarkan oleh CKEditor untuk membantu membangun aplikasi. CKEditor sendiri merupakan sebuah server untuk melakukan integrasi dari server ke server [9]. Salah satu contohnya dalam *comment methods* dapat digunakan untuk sinkronisasikan *comment* antar Ckeditor Cloud dengan service yang lainnya [9]. Dengan tambahan lainnyam, CKEditor *cloud service* dapat digunakan sebagai *database* untuk menampung komen komen didalamanya, dan juga data data yang diambil menggunakan dalam CKEditor *cloud service* ini memungkinkan untuk di *download*.

Untuk *Authentication* pada CKEditor *Cloud Service* membutuhkan hal khusus sebagai *identifier header* yang bernama HMAC (HTTP *Authorization Scheme*). Terdapat dua *header* yang dibutuhkan oleh CKEditor sebagai *identifier* dalam melakukan *request* data ke server.



Gambar 3. 12 *Header* HMAC

Untuk menjalankan sesuai kebutuhan CKEditor, yaitu dengan membuat sebuah fungsi untuk membuat *signature* yang dibutuhkan. Berikut merupakan contoh pembuatan *signature* yang dibutuhkan.

```python
import hmac
import hashlib
import json
import urllib.parse

def hmacDigest(data, key):
    keyEncoded = key.encode()
    dataEncoded = data.encode()

    h = hmac.new(keyEncoded, dataEncoded, hashlib.sha256)

    return h.hexdigest()


def generateSignature(apiSecret, method, uri, timestamp, body):
    url = urllib.parse.urlparse(uri)
    path = url.path

    if (url.query):
        path = path + "?" + url.query

    methodUpperCase = method.upper()
    data = methodUpperCase + path + str(timestamp)

    if (body):
        data += json.dumps(body, separators=(',',':'))

    return hmacDigest(data, apiSecret)
```

Gambar 3. 13 *Function* untuk *signature*

Pada *function* ini membutuhkan 4 buah *library* yaitu hmac, hashlib, json dan urlibparse. Dan membutuhkan 2 buah *function* yang dibutuhkan untuk melakukan generasi *signature* dan hmac digest untuk generasi *hmac*. Nantinya akan digunakan dengan *encode* data dan *api secret key* yang dapat didapatkan.

Gambar 3. 14 Contoh implementasi *signature*

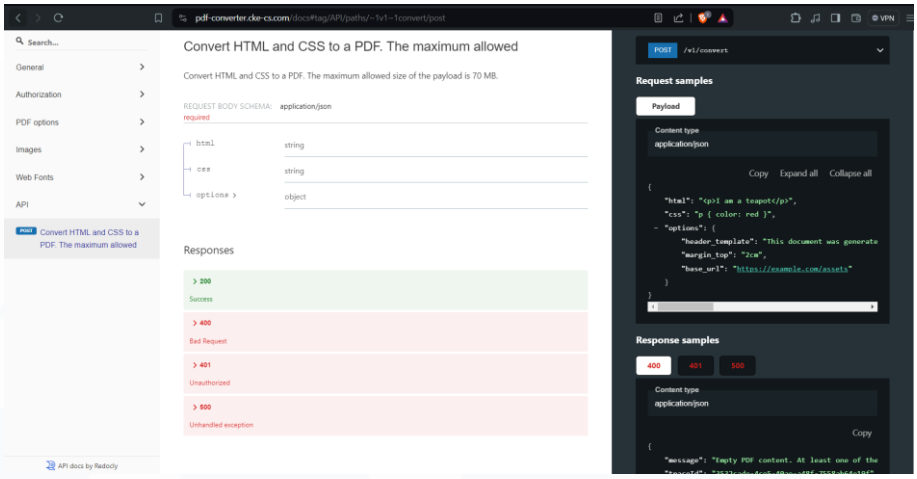
Pada gambar akan menghasilkan *true* jika token yang di*generate* sama dengan token yang disediakan. Namun pada saat pembuatan langsung, tanggal / timestamp harus diambil secara *realtime*. Pada CKEditor *cloud service* terdapat metode yang dapat digunakan oleh *developer* sebagai perantara antara *service* CKEditor, berikut merupakan list dari metode:

- Collaboration
- Documents
- Revisions
- Storage
- Comments
- Suggestion
- Environtments
- Users
- Editor Bundles
- Statistic

Masing masing dari metode memiliki manfaatnya masing masing.

3. HTML to PDF Converter API

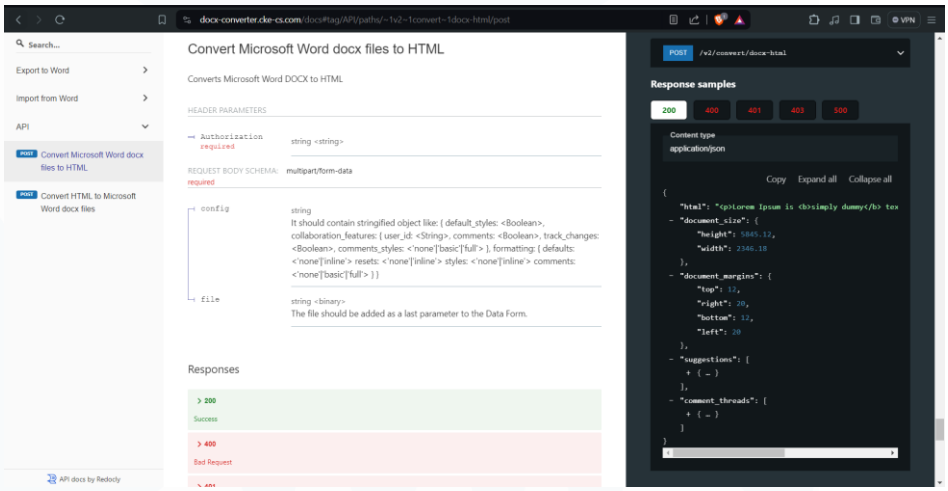
Endpoint ini digunakan untuk melakukan *convert* HTML dan CSS file ke PDF sehingga segala bentuk yang terdapat *text editor* dapat dibuat ke file PDF.



Gambar 3. 15 API HTML dan CSS to PDF

4. HTML to DOCX Converter API

Endpoint ini digunakan untuk melakukan *convert* HTML dan CSS file ke Word sehingga segala bentuk yang terdapat *text editor* dapat dibuat ke file Word.



Gambar 3. 16 API HTML dan CSS to Word

Untuk membuat aplikasi *backend* yang dibutuhkan sesuai dengan *service* yang dibutuhkan oleh CKEditor mahasiswa perlu membuat *custom endpoint* untuk menjadi penghubung *service* yang dimiliki oleh CKEditor dengan *service* yang akan dibuat secara

native. Berikut merupakan contoh *enpoint* yang dibuat oleh mahasiswa.

```python
# Recent
@ckbox_router.get("/assets/recent/}", tags=['CKBOX - Recent'])
async def get_recent_assets(
    authorization: str = Header(None),
    categoryId: str | None = Query(None, description="Identifier of the assets category."),
    offset: int | None = Query(0, description="The number of assets to skip before starting to collect the result set."),
    limit: int | None = Query(50, ge=1, le=500, description="The numbers of assets to return."),
    order: str | None = Query("desc", enum=["asc", "desc"], description="Return assets in ascending or descending order of
    workspaceId: str | None = Query(None, description="Identifier of the workspace."),
):
    jwt = authorization.split(" ")[1] if authorization else None
    response_get_recent_assets = ckbox_service.get_recent_assets(jwt, categoryId, offset, limit, order, workspaceId)
    return JSONResponse(response_get_recent_assets)

@ckbox_router.put("/assets/recent/{workspaceId}", tags=['CKBOX - Recent'])
async def update_recent_assets(request: Request, workspaceId:str | None = None, authorization: str = Header(None)):
    jwt = authorization.split(" ")[1] if authorization else None
    request_body = await request.json()
    response_update_recent_assets = ckbox_service.update_recent_assets(workspaceId, jwt, request_body)
    return JSONResponse(response_update_recent_assets)

# Assets Metadata
@ckbox_router.patch("/assets/{assetsId}/metadata", tags=['CKBOX - Assets Metadata'])
async def update_metadata_assets(assetsId: str, request: Request, authorization: str = Header(None)):
    jwt = authorization.split(" ")[1] if authorization else None
    request_body = await request.json()
    response_update_metadata_assets = ckbox_service.update_metadata_assets(assetsId, request_body, jwt)
    return JSONResponse(response_update_metadata_assets)
```

Gambar 3. 17 Contoh *endpoint* CKBOX

Pada *code* yang diberikan dalam Gambar 3,17, terdapat 3 buah *enpoint* yang dibuat yang masing masing dari *endpoint* memiliki *service* masing masing. *Service* tersebut didefinisikan sebagai "ckbox_service" pada *code* . Pada setiap *endpoint* memiliki *query parameter / path parameter* yang disimpan dalam variable yang berbeda. Namun harus tetap sesuai dengan spesifikasi yang diminta oleh CKEditor. Pada *endpoint* pertama yaitu "/assets/recent/" terdapat 1 (satu) buah *Header* dan 5 (lima) buah *query parameter*. *Header* yang dimiliki memiliki variabel "authorization" yang didefinisikan sebagai *string* dan *required*.

Berikut adalah penjelasan mengenai parameter yang diterima oleh endpoint ini

- authorization: Sebuah header yang digunakan untuk otorisasi, berisi token JWT (*JSON Web Token*).

- categoryId: Berisifat opsional, digunakan untuk mengidentifikasi kategori aset tertentu.
- offset: Menentukan jumlah aset yang dilewati sebelum mulai mengumpulkan hasil, dengan nilai *default* 0.
- limit: Menentukan jumlah aset yang akan dikembalikan, dengan batas minimum 1 dan maksimum 500, dan nilai *default* 50.
- order: Menentukan urutan pengembalian aset berdasarkan properti *lastUsedAt*, bisa dalam urutan *ascending* (asc) atau *descending* (desc), dengan nilai *default* "desc".
- workspaceId: Bersifat opsional, digunakan untuk mengidentifikasi workspace tertentu.

Di dalam fungsi tersebut juga, token JWT diambil dari header *authorization*. Kemudian, fungsi "get_recent_assets" dari *service* akan dipanggil dengan parameter yang diterima dari *user*. Hasil dari *return service* tersebut kemudian dikembalikan sebagai respon dalam format JSON menggunakan JSONResponse yang dimiliki FastAPI. Berikut merupakan *code* dari *service* yang dibutuhkan oleh *endpoint* "/assets/recent/".

```python
def get_recent_assets(jwt: str, categoryId: str = None, offset: int = 0, limit: int = 50, order: str = "desc", workspaceId: str = None):
    recent_assets_path = "assets/recent"
    ckbox_api_url = get_ckbox_url(recent_assets_path)
    jwt = generate_ckbox_jwt(jwt)
    headers = {
        "Authorization": jwt
    }
    params = {
        "categoryId": categoryId,
        "offset": offset,
        "limit": limit,
        "order": order,
        "workspaceId": workspaceId
    }
    try:
        response = requests.get(ckbox_api_url, headers=headers, params=params)
        return {"data":response.json(), "status": "success", "status_code": response.status_code}
    except Exception as e:
        return f"Error: {e}"
```

Gambar 3. 18 *Service* yang menjalankan API

Function digunakan oleh *enpoint* untuk menjadi perantara antara CKEditor *service* dengan aplikasi yang dibuat ini. Pada code dapat dibagi menjadi 5 *step:*

1. *Function Parameter*:

   Berikut merupakan parameter yang diaambil oleh *function* sehingga dapat menjalankan prosesnya dengan baik.

   - jwt: Token JWT yang digunakan untuk *authorization*.
   - categoryId: berisfat opsional, ID kategori aset yang ingin diambil.
   - offset: *Default* 0, jumlah aset yang dilewati sebelum mulai mengumpulkan hasil.
   - limit: *Default* 50, jumlah aset yang akan dikembalikan, dengan batas minimum 1 dan maksimum 500.
   - order: *Default* "desc", urutan pengembalian aset berdasarkan properti lastUsedAt (*ascending* atau *descending*).
   - workspaceId: Opsional, ID *workspace* tertentu.

2. Membuat *url* dan *header*:

   Fungsi dari step ini yaitu dengan membuat *url* untuk melakukan http *request* kepada *service* yang dimiliki oleh CKEditor dengan *header* yang sudah diparse kedalam bentuk variabel sehingga dapat diterima dengan baik dan menghasilkan *response* yang baik. Berikut poin-poinnya:

   - recent_assets_path: String yang menunjukkan path relatif untuk endpoint aset terbaru.
   - ckbox_api_url: URL lengkap untuk endpoint dengan memanggil fungsi get_ckbox_url dan memberikan recent_assets_path.

- jwt: Token JWT yang diproses dengan memanggil fungsi generate_ckbox_jwt.
- headers: Header HTTP yang berisi token JWT dalam bentuk Authorization.

3. Membuat *Query Parameter:*

   Query Parameter dibutuhkan dalam sebuah *request* kepada *service* CKEditor. Dengan adanya *query parameter* dapat menjadi sebuah *filter* yang dapat dilakukan dalam request yang nantinya akan dikirim, berikut merupakan variabel yang diambil:

   - params: *dictionary* yang berisi *query parameter* untuk *request* dengan metode GET, termasuk categoryId, offset, limit, order, dan workspaceId sebagai *query parameter* yang dibutuhkan.

4. HTTP *Request* Ckeditor *service:*

   Untuk melakukan request pada bahasa Python dapat menggunakan sebuah *library* yang bernama request yang berfungsi untuk men-*initialize* sebuah *request*. Menggunakan requests.get ke URL CKBOX dengan *header* dan *query parameter* yang telah dibuat dalam bentuk variabel sebelumnya. Lalu, jika *request* yang dilakukan berhasil, *response* JSON dari *server* CKBOX di-*return* dengan struktur sebagai berikut {"data": response.json(), "status": "success", "status_code": response.status_code}.

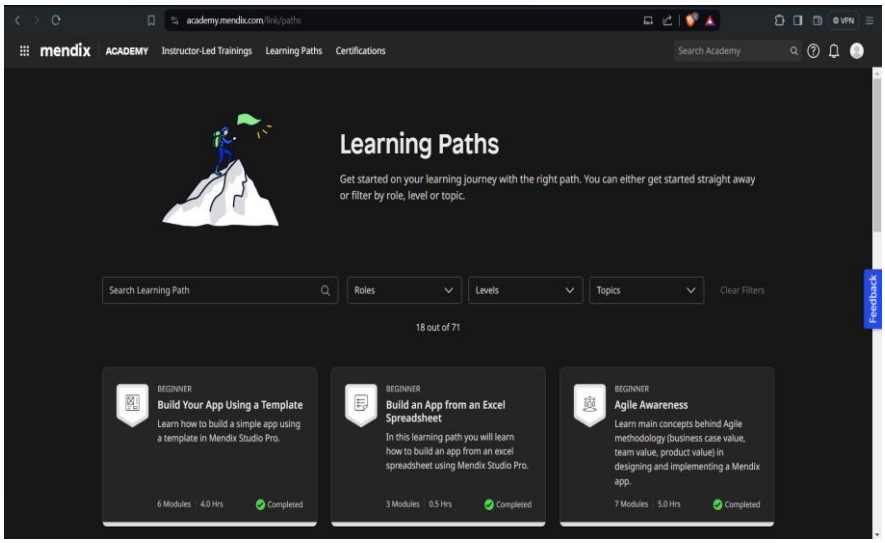
5. *Error Handling*:

   Namun, jika terjadi kesalahan selama *request* atau proses yang dijalankan sebelumnya mengalami *error*, dengan adanya *error* handling dapat menangkap kesalah yang terjadi dan mengembalikan pesan kesalahan dalam format string.

Secara keseluruhan, fungsi ini bertanggung jawab untuk berkomunikasi dengan API CKBOX, mengirim permintaan GET, dan mengembalikan hasil dalam bentuk JSON atau pesan kesalahan jika terjadi kegagalan.

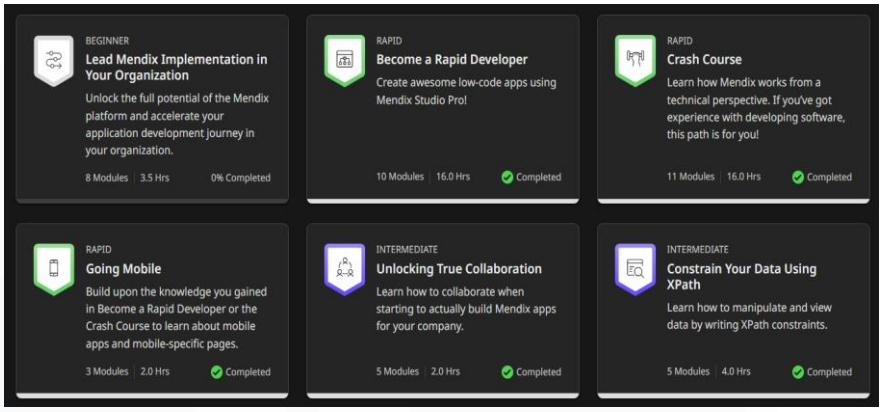
2. **Mendix**

Pada kesempatan magang ini mahasiswa memperoleh satu buah sertirfikat mendix, sebagai *Rapid Developer*. Sertifikat ini dapat diambil oleh mahasiswa jika telah menyelesaikan *course* yang disediakan dalam *website academy* mendix. *Academy* mendix memiliki berbagai macam *course* yang berguna dalam melakukan *best practice* selama men-*develop* aplikasi dengan *tools* mendix. *Academy* Mendix juga memiliki beberapa *level* sesuai dengan kemampuan atau proses selama mahasiswa mengerjakan *course* pada *website academy* mendix, berikut merupakan tampilan yang dimiliki oleh *academy* mendix.



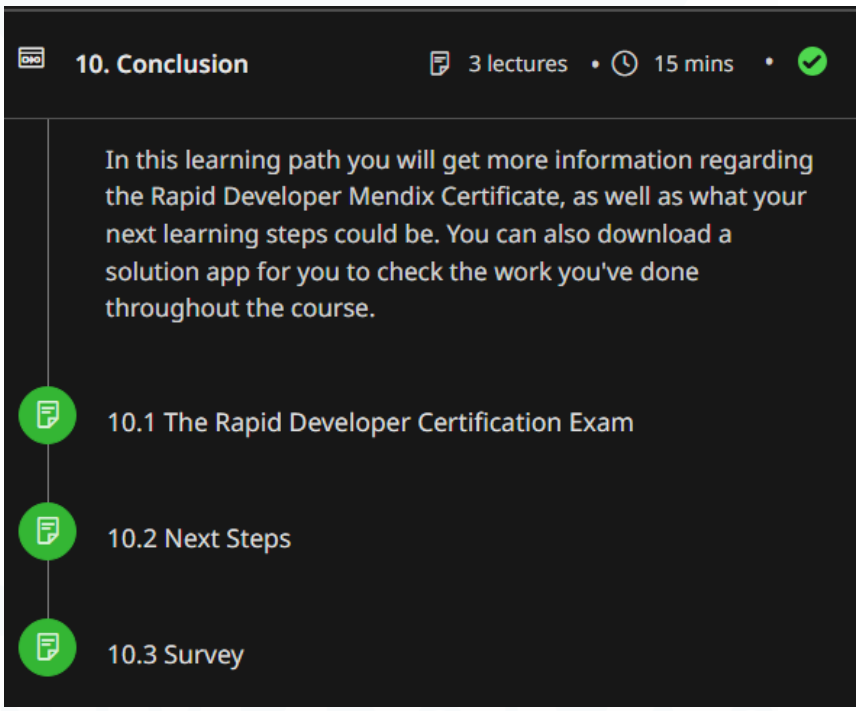
Gambar 3. 19 Tampilan *academy* mendix

Untuk sertifikasi yang dapat diambil jika telah menyelesaikan seluruh *Rapid Course* terutama *Become a Rapid Developer*.



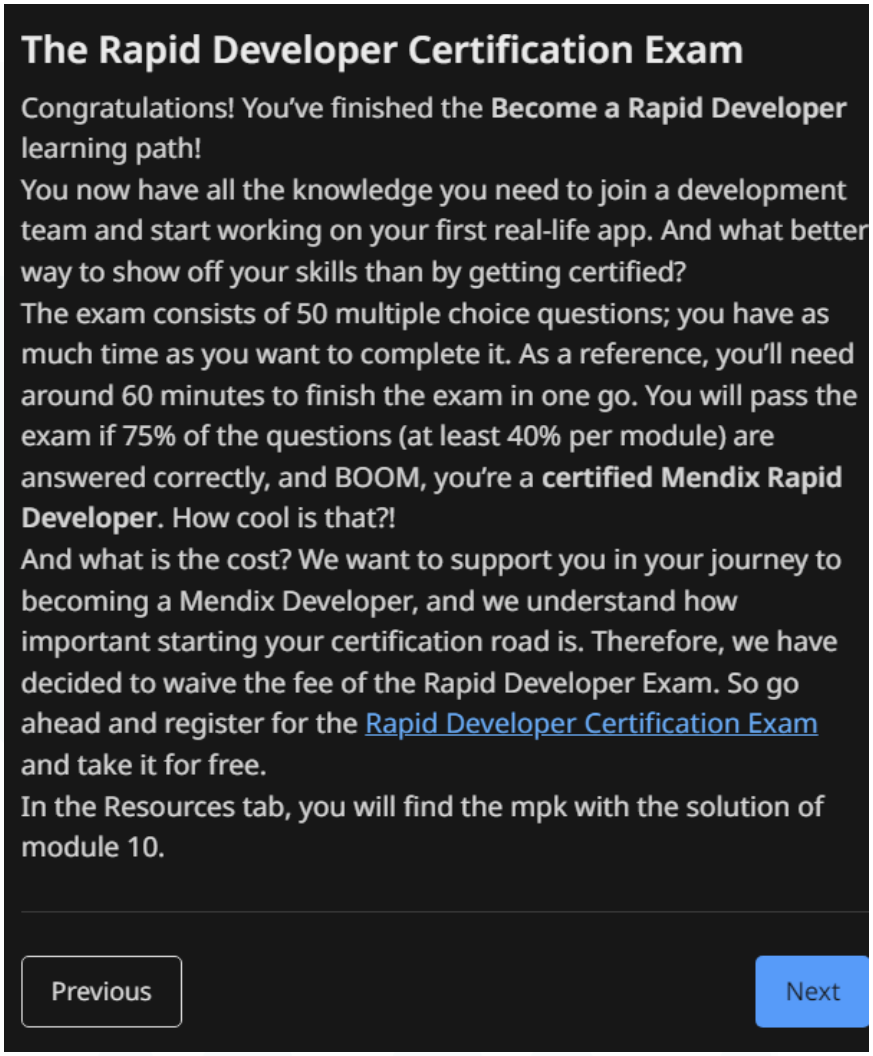
Gambar 3. 20 Tampilan *Rapid Course*

Lalu sertifikat dapat diambil setelah mahasiswa daftar untuk mengikuti *exam* yang dibuat oleh mendix. *Exam* dapat diambil dalam *Become a Rapid Developer*.



Gambar 3. 21 Tampilan bagian 10 *Become a Rapid Developer*

Dan selanjut dapat mendaftar *exam* dalam tampilan *link* berikut.



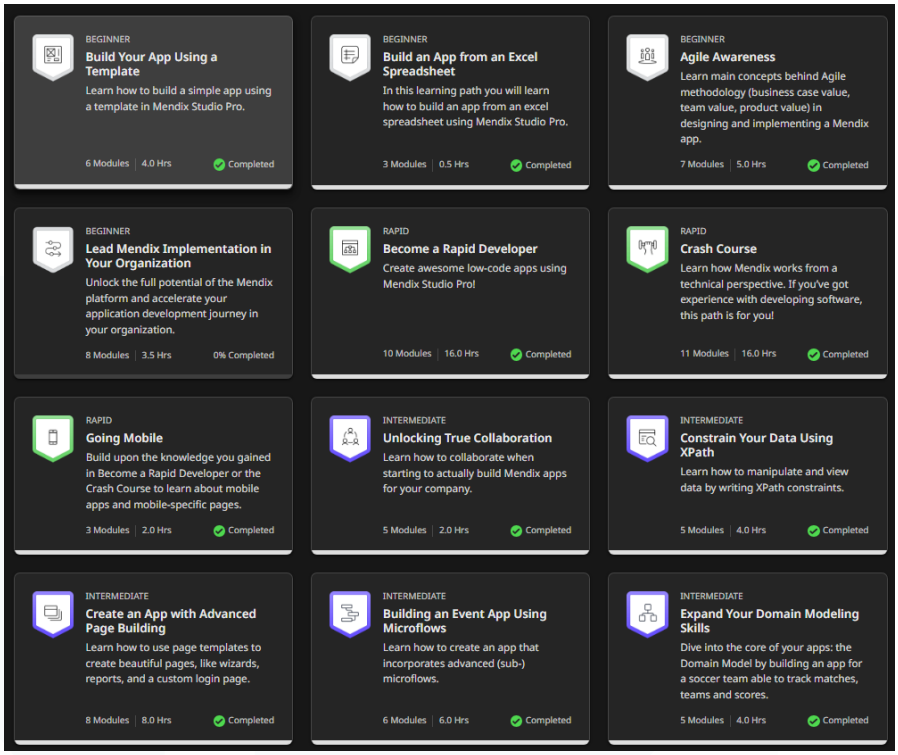
Gambar 3. 22 Pendaftaran Mendex *Rapid Developer*

Setelah menyelesaikan *esxam* tersebut sertifikasi akan dikirim melalui email.

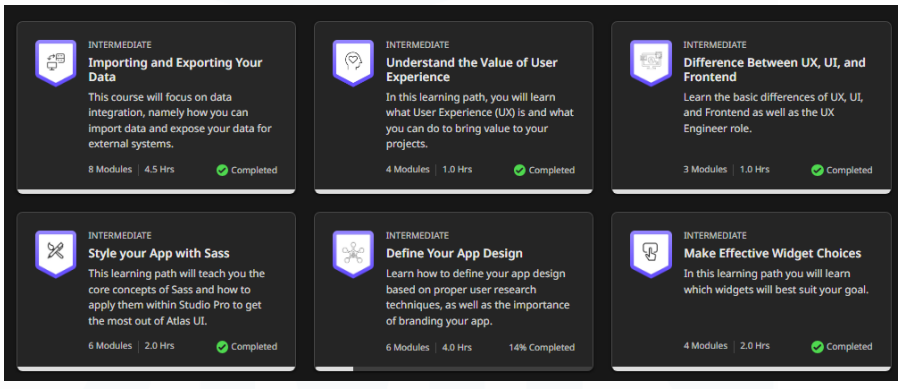
Gambar 3. 23 Bukti Sertifikasi Mahasiswa

Secara keseluruhan berikut merupakan *list course* yang telah diselesaikan oleh mahasiswa:

- *Build Your App Using Template*
- *Build an App from ann Excel Spreadsheet*
- *Agile Awareness*
- *Become a Rapid Developer*
- *Crash Course*
- *Going Mobile*
- *Unlocking True Collaboration*
- *Expand Your Domain Modelling Skill*
- *Importing and Exporting Your Data*
- *Understand the Value of User Experience (UX)*
- *Difference Between UX,UI and Frontend*
- *Style Your App with Sass*
- *Make Effective Widget Choices*
- *Build an Pluggable Widget*



Gambar 3. 24 *Course list* 1



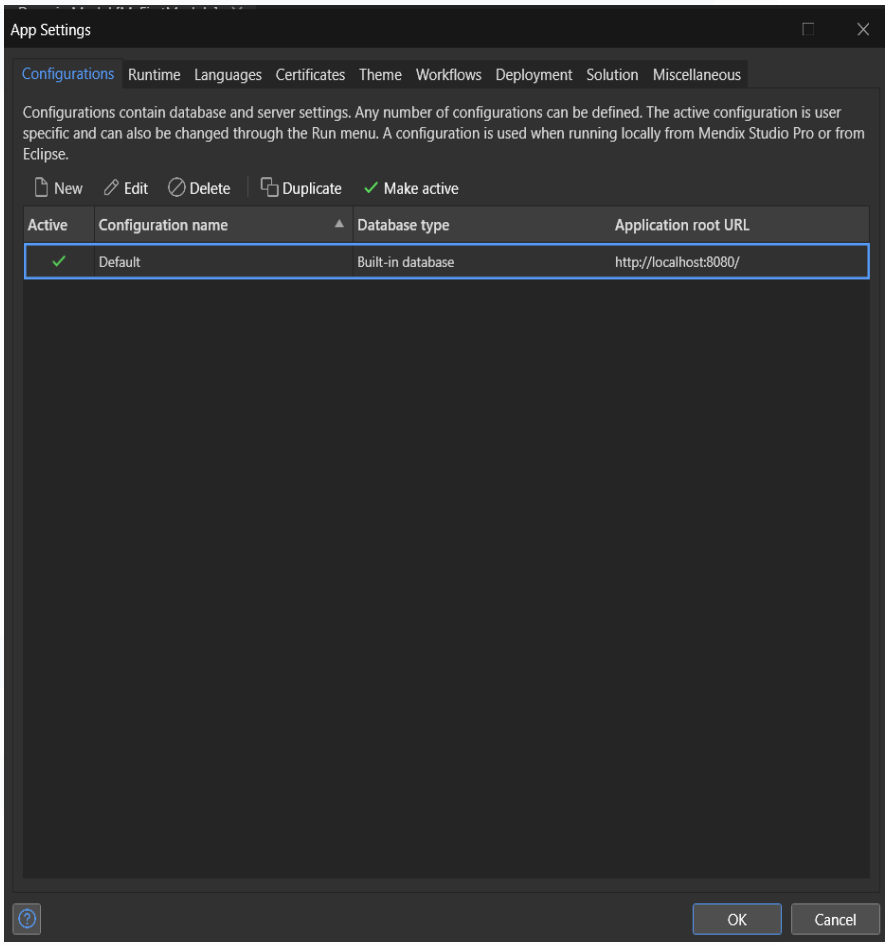
Gambar 3. 25 *Course list 2*

a. Mendix Studio Pro

Mendix Studio Pro merupakan sebuah software yang dapat digunakan untuk membuat aplikasi *Low Code Application*.

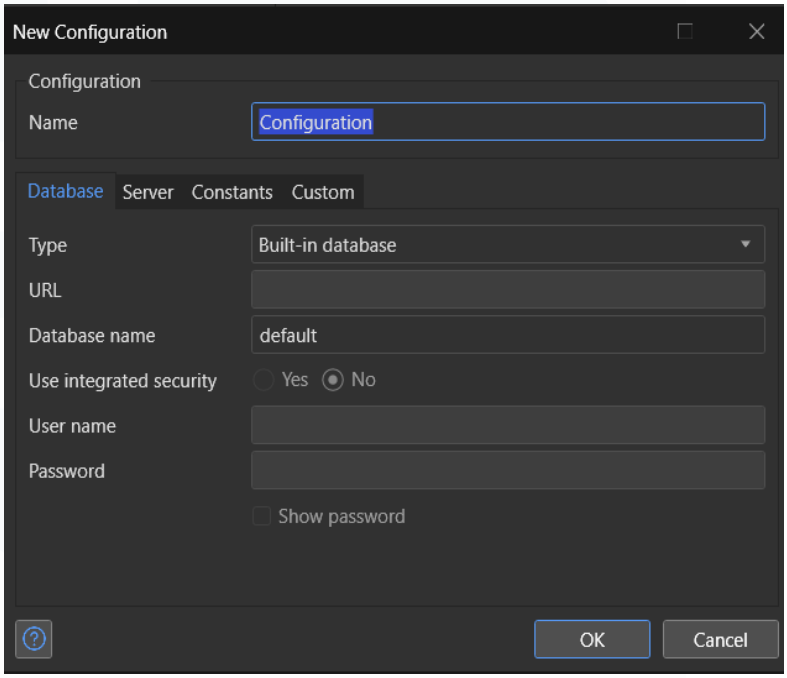
Gambar 3. 26 Logo Mendix Stduo Pro (Old)

Dengan menggunakan Mendix Studio Pro user dapat membuat aplikasi dengan cepat dan mudah (*drag and drop)*. Mendix sendiri memiliki *Local Databasei* sehingga tidak perlu membuat *database* external.



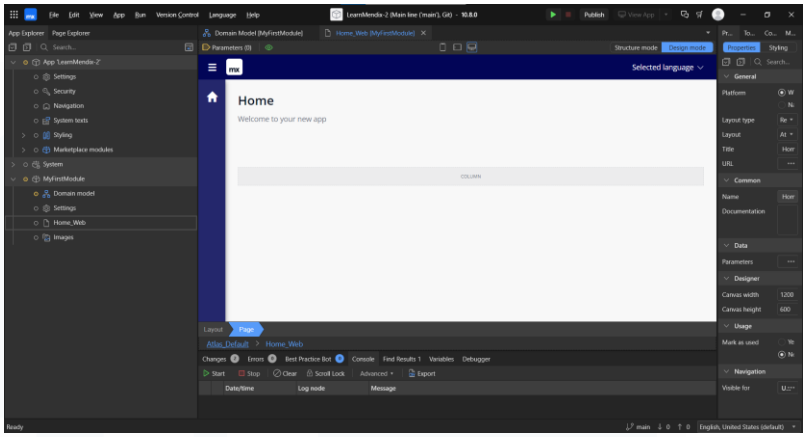
Gambar 3. 27 Tampilan *database selector* pada Mendix

Dengan adanya ini tentunya dapat membuat developer lebih mudah dalam membuat sebuah aplikasi, dikarenakan terdapat fleksibelitas dalam melakukan *develop* sebuah aplikasi. Jika ingin memakai *database* external dapat melakukanya dengan menginput koneksi *database* tersebut.



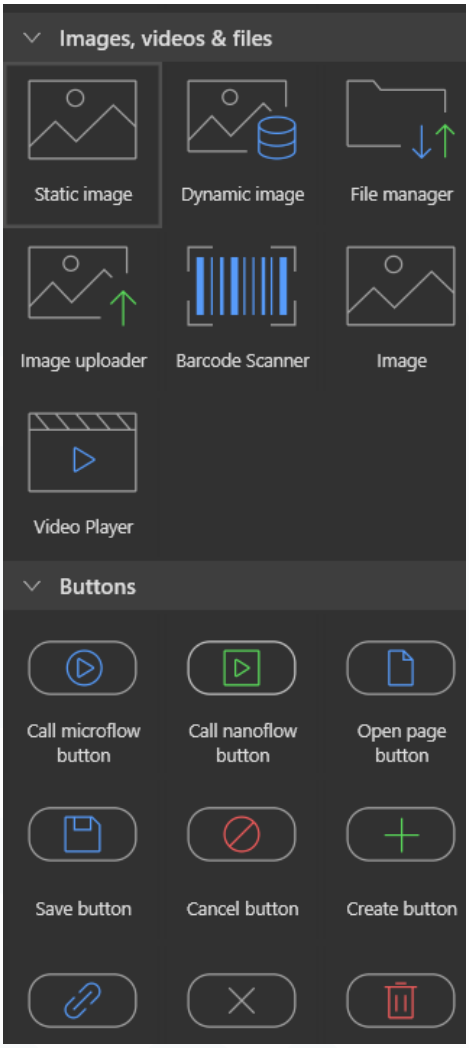
Gambar 3. 28 Memakai *database* external

Dengan mengunakan mendix dapat melihat *design* yang telah dibuat dengan memilih *design mode* sehingga *user* dapat mengetahui tampilan website berdasarkan *component* yang digunakan oleh *user*.



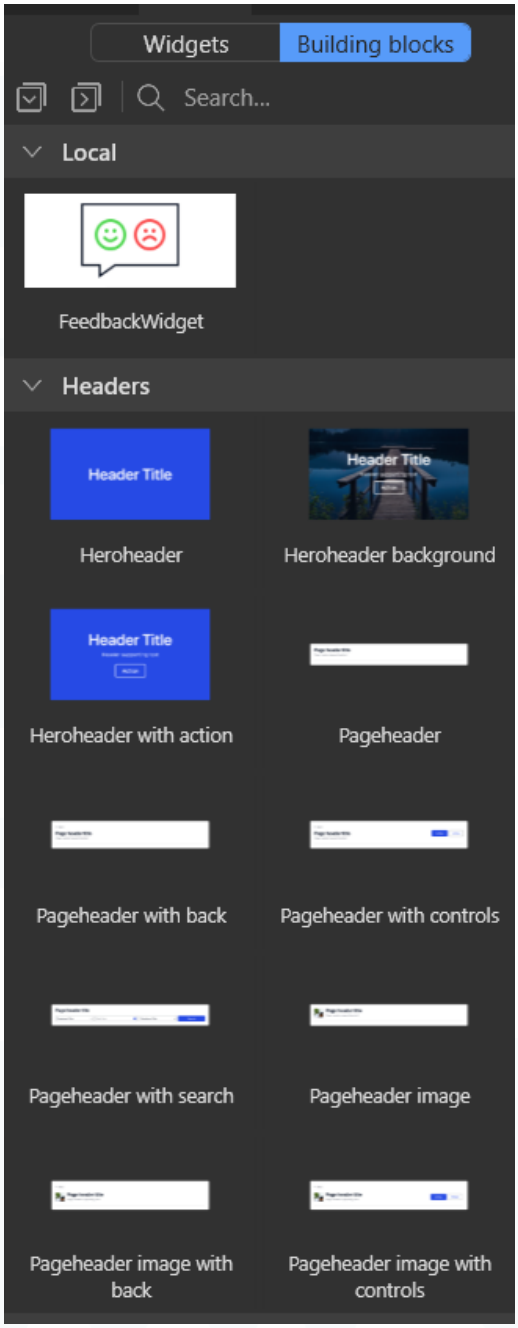
Gambar 3. 29 Tampilan *default* Mendix *Application*

Design mode dapat berguna oleh *developer* dikarenakan dapat membuat melakukan *inspect* dalam aplikasi yang dibuat. Mendix sendiri menyediakan banyak komponen/*widget* seperti *button*, *chart*, *input element*, *data container*, dll.



Gambar 3. 30 Contoh Widget

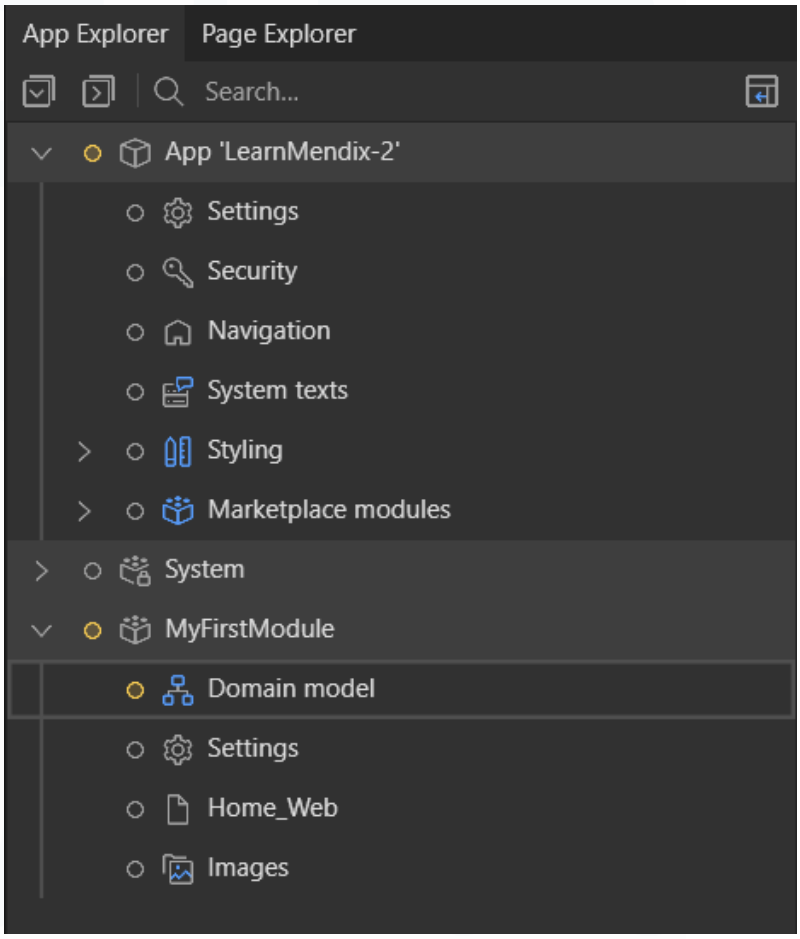
Mendix juga menyediakan beberapa *building block* yang dapat gunakan oleh user dalam membuat aplikasi. Dalam menggunakan Mendix user juga dapat mengatur *layout responsive* agar dapat diakses dengan baik dalam segala *device* yang mengakses *website.*



Gambar 3. 31 Contoh *Building Block*

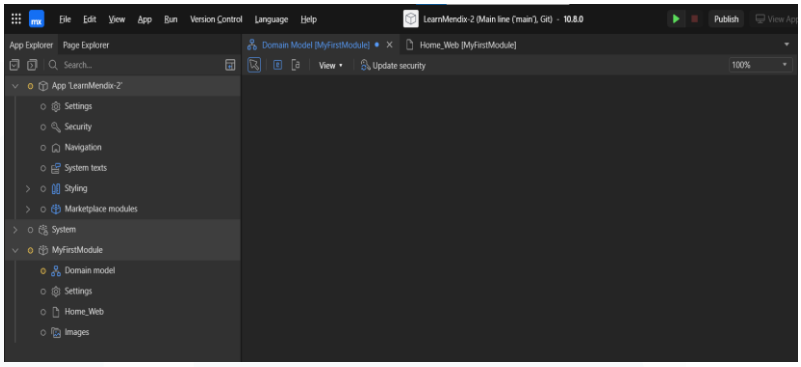
Gambar 3.31 menunjukan komponen komponen apa saja yang dapat digunakan dalam pembuatan aplikasi mendix, dengan adanya *building block* user tidak perlu lagi untuk membuat komponen dari awal. Untuk mengatur data data

atau *table* yang akan digunakan dalam aplikasi, dalam mendix terdapat *domain model*. *Domain Model* hampir sama seperti RDBMS, dengan menggunakan *Domain Model* pada mendix *user* dapat langsung mengatur *flow table* yang akan digunakan pada aplikasi. *Domain Model* dapat ditemukan dalam *App Explorer* pada *interface* Mendix Studio Pro.



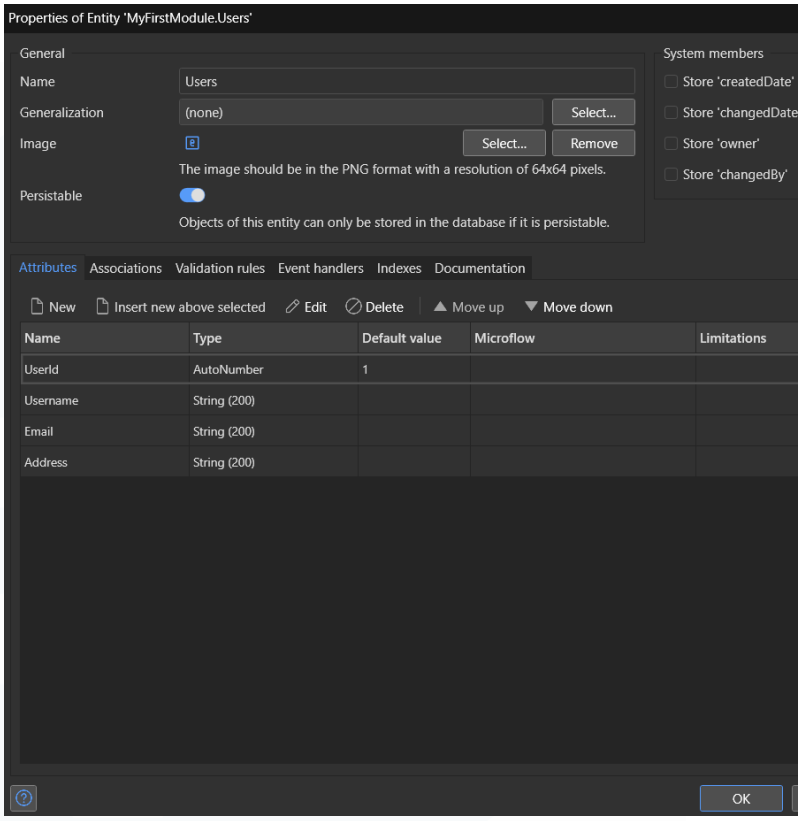
Gambar 3. 32 Domain Model (1)

Gambar 3.32 menunjukan struktur proyek yang berada pada masa develop aplikasi Mendix. User dapat mengatur seperti apa *data* yang akan digunakan, hanya dengan melakukan input dan melakuakn *drag and drop*.



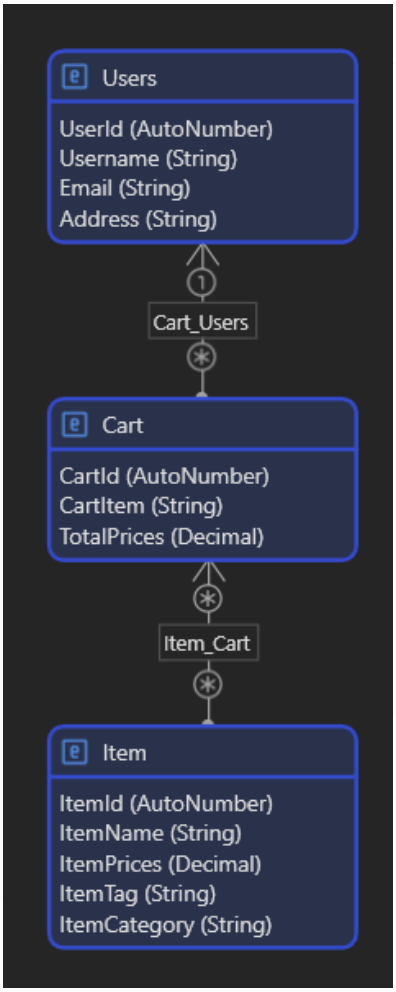
Gambar 3. 33 *Domain Model (2)*

Pada Gambar 3.33 merupakan contoh awal dari bagian *domain model* awal saat aplikasi baru dibuat atau inisialisasi. Tampilan awal yang didapat saat baru membuat ingin membuat domain model yaitu seperti gambar . Setelah mendapat tampilan seperti yang ada , *user* hanya harus melakukan *table* yang ingin dibuat.



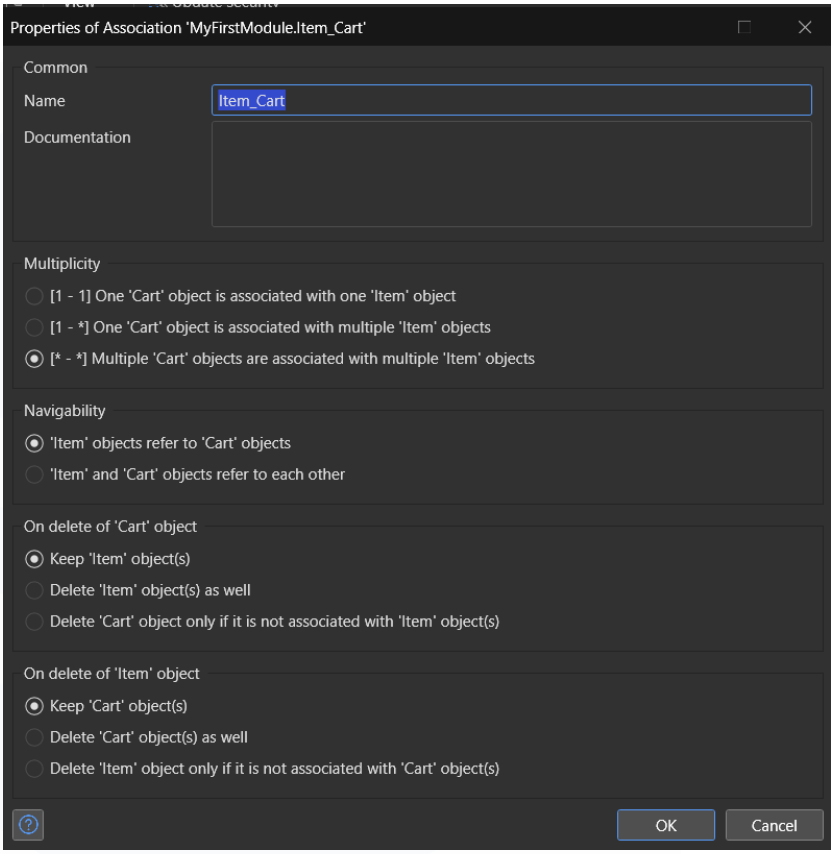
Gambar 3. 34 Tampilan Input Entity

Setelah melakukan input user dapat membuat beberapa *table*, yang nantinya untuk menghubungkan antar *table* tidak memerlukan adanya *Foreign Key* pada *Domain Model user* hanya memerlukan untuk menarik garis antar *entity* seperti gambar dibawah ini.



Gambar 3. 35 Relasi antar *Entity Domain Model*

Pada Gambar 3.35 menunjukan contoh relasi yang berjalan pada Mendix, dengan menggunakan relasi tersebut user tidak perlu lagi menggunakan *foreign key* agar dapat mengakses *table* yang berkaitan. Untuk dapat mengatur relasi antar *entity* dapat dilakukan dengan cara *double click* pada relasi yang terlihat dan akan muncul *menu* yang bertujuan untuk mengatur relasi tersebut.



Gambar 3. 36 Tampilan menu mengtur relasi

Berdasarkan pada Gambar 3.36. dapat dilihat menu tersebut *user* dapat mengatur relasi hanya dengan memilih *option* yang terdapat pada menu untuk mengatur relasi antar *entity* seperti gambar . Dengan hal hal *user* dapat membuat aplikasi CRUD (*Create Read Update Delete*) dengan baik.

3. Contract Management

Proyek ini merupakan sebuah proyek yang dibuat untuk mempermudah proses input data pada sebuah *contract* yang berjalan pada sebuah *web application*. Teknologi yang digunakan pada proyek ini meliputi:

- Vue: Vue merupakan sebuah *frontend framework* yang digunakan untuk membuat tampilan *user*. Versi yang

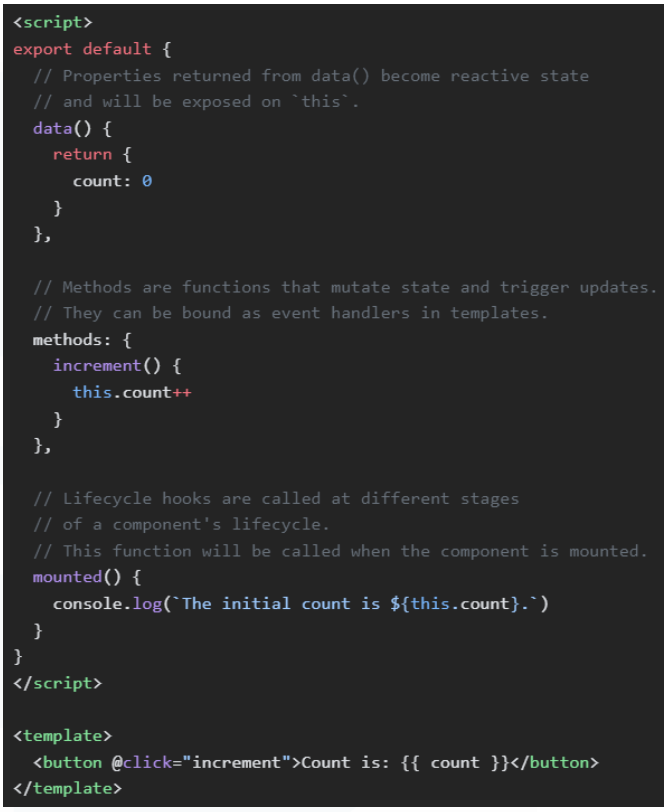
digunakan dalam proyek ini yaitu Vue3, Vue3 merupakan versi terbaru yang dimiliki oleh Vue.

Gambar 3. 37 Logo Vue

Penggunaan Vue dapat dibagi menjadi 2 (dua) yaitu:

- *Option API*:

  Penggunaan *Option API* merupakan cara lama yang dapat digunakan didalam *framework* VUE. Dengan menggunakan ini *code* harus lebih terstruktur berdasarkan *method* yang digunakan [9]. Berikut merupakan contoh penggunaan *Option API*.



Gambar 3. 38 *Option API*

Pada Gambar 3.38 menunjukan *code* dapat disimpulkan penggunaan *Option API* yaitu *class based* atau berbasis *class*. Dengan seperti ini segala variabel ataupun *property* yang terdapat pada sebuah *file option api* harus diakses dengan "*this*" dengan begitu *property* yang dimiliki dapat diakses dalam sebuah method

- *Composition API*

  Lalu, pada *composition api* struktur *code* menjadi lebih *functional*. Segala bentuk *code* yang dibuat dalam *composition api* hampir sama seperti penggunaan Javascript pada umumnya.

Berikut merupakan contoh penggunaan *Composition API* dalam Vue.

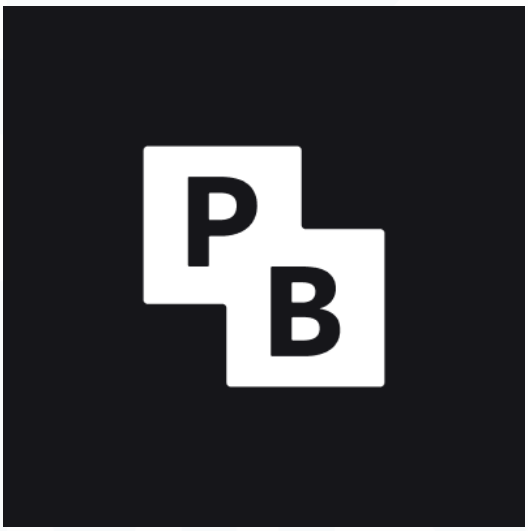
Gambar 3. 39 Contoh *Composition API*

Pada Gambar 3.39 menunujukan contoh penggunaan *Composistion API* yang dimiliki oleh *framework* VUE. Dengan penggunaan *compostion api*, script pada javascript ditulis dengan "*script setup"*. Atribut *setup* disini berfungsi dalam meningkatkan performa *compiling time* sehingga dapat digunakan. Dengan menggunakan *composition api* user dapat membuat aplikasi tanpa perlu menulis ulang *code* berulang kali [9].

Pada proyek ini mahasiswa menggunakan *composition api* dikarenakan sesuai *best practice*, dengan membuat *full* aplikasi menggunakan VUE sangat disarankan dengan menggunakan *composition api* [9]. Tetapi, dengan memilih *composition api* sangat memungkinkan

untuk user dapat menggunakan *option api* dibeberapa *case* yang dibutuhkan [9].

- FastApi: Untuk *framework* yang digunakan dalam membuat aplikasi *backend* mahasiswa menggunakan *FastAPI* dikarenakan dengan menggunakan *FastAPI* dapat membuah sebuah *endpoit* dan *service* dengan cepat lalu ditambah dengan dokumentasi yang lengkap maka dari itu pilihan *FastAPI* menjadi pilihan yang tepat dalam membuat sebuah *backend application* yang baik.
- PocketBase: Pocketbase merupakan sebuah *opensource SaaS (Software as a Service)* yang dapat sangat membantu dalam pembuatan aplikasi [10].



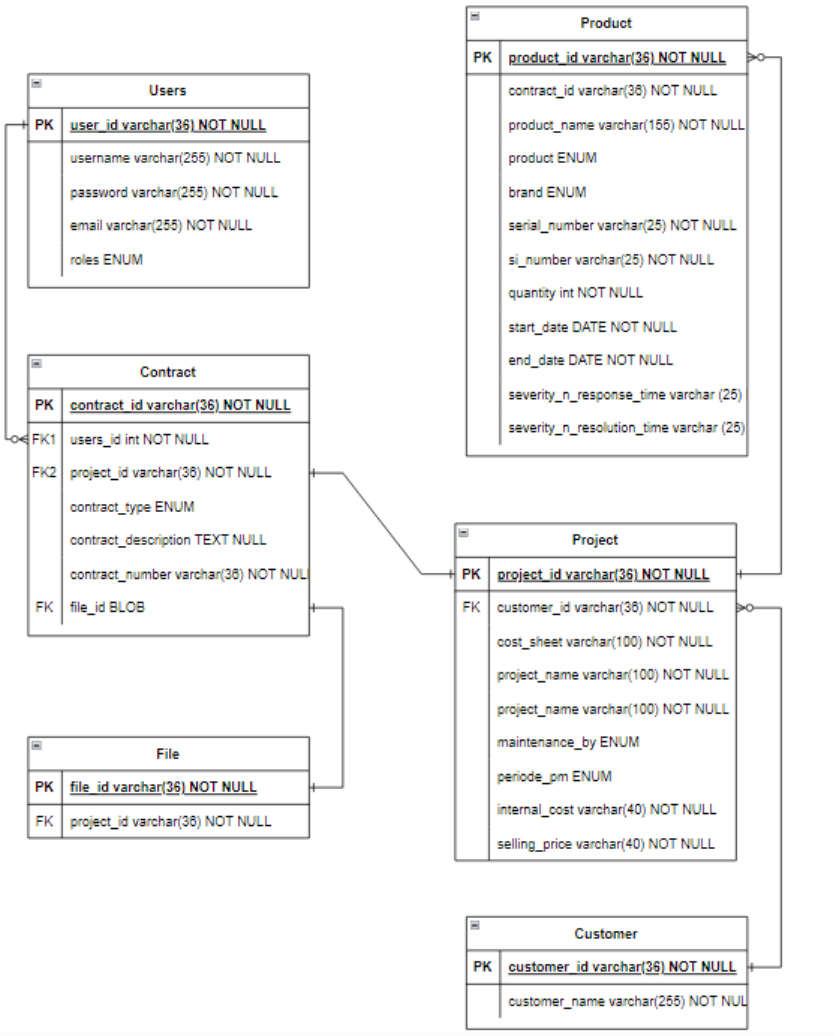
Gambar 3. 40 Logo PocketBase

Dalam proyek ini PocketBase digunakan untuk menyimpan data *user* yang dapat mengakses *web*. Dan dapat membantu proses *developi* dikarenakan proses *authentication* sudah dapat dilakukan oleh pocketbase. Selain dapat mengatur *authentication* pocketbase juga

sangat cocok dalam menyimpan *file* namun dengan penyimpanan terbatas, tetapi dapat *connect* dengan platform *cloud* lainnya seperti AWS, GCloud, dll.

- Postgress Database: Untuk *database* yang digunakan dalam membuat aplikasi ini yaitu dengan menggunakan *database*, dikarenakan postgres *database*.

Mahasiswa membangun aplikasi *contract management* dengan menbuat API untuk menyediakan data kepada *frontend*. Mahasiswa juga merancang struktur *database* yang digunakan dalam aplikasi. Pada Gambar 3.41 dibawah ini merupakan contoh dari gambaran struktur *database* yang digunakan dalam aplikasi.

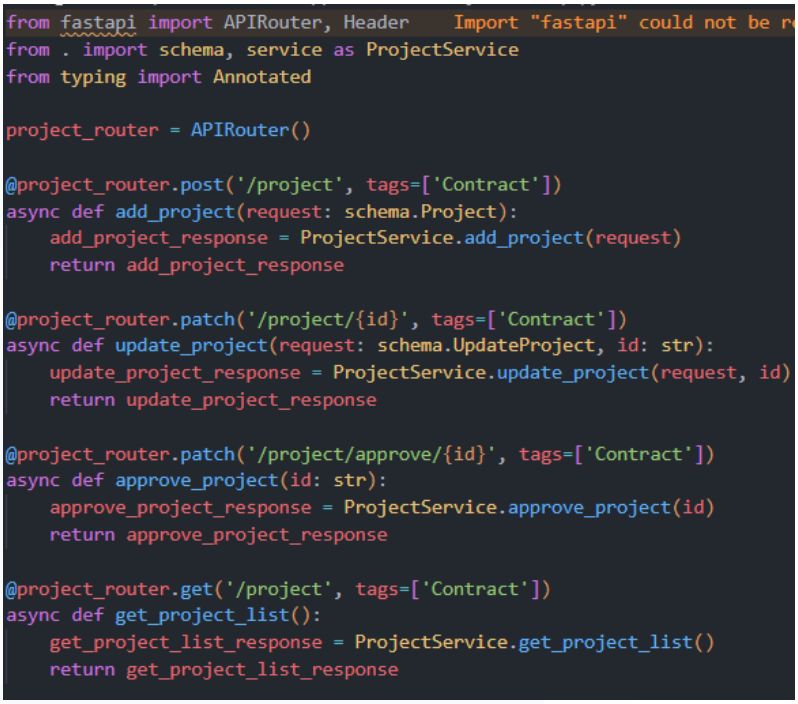


Gambar 3. 41 Struktur *database* aplikasi

Selanjutnya mahasiswa membangun API dan *service* yang diperlukan. Pada aplikasi *backend* ini terdapat beberapa *service* besar yang dibangun:

- Contract

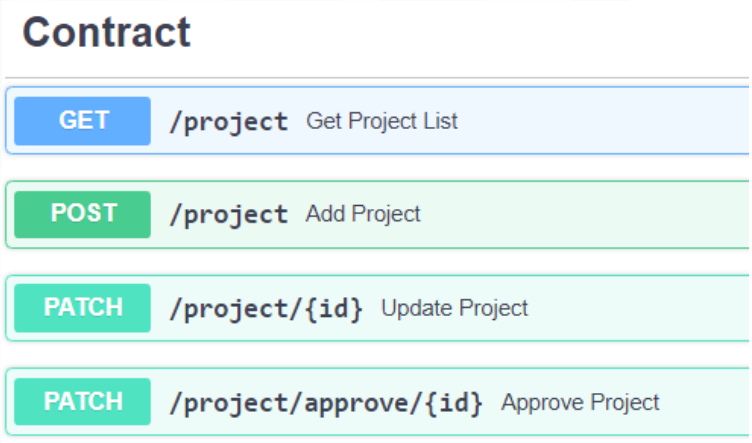
  Service contract dibuat untuk mengatur beberapa *service* yang bertujuan untuk membuat proses yang berkaitan dengan *contract*.



```python
from fastapi import APIRouter, Header    Import "fastapi" could not be r
from . import schema, service as ProjectService
from typing import Annotated

project_router = APIRouter()

@project_router.post('/project', tags=['Contract'])
async def add_project(request: schema.Project):
    add_project_response = ProjectService.add_project(request)
    return add_project_response

@project_router.patch('/project/{id}', tags=['Contract'])
async def update_project(request: schema.UpdateProject, id: str):
    update_project_response = ProjectService.update_project(request, id)
    return update_project_response

@project_router.patch('/project/approve/{id}', tags=['Contract'])
async def approve_project(id: str):
    approve_project_response = ProjectService.approve_project(id)
    return approve_project_response

@project_router.get('/project', tags=['Contract'])
async def get_project_list():
    get_project_list_response = ProjectService.get_project_list()
    return get_project_list_response
```

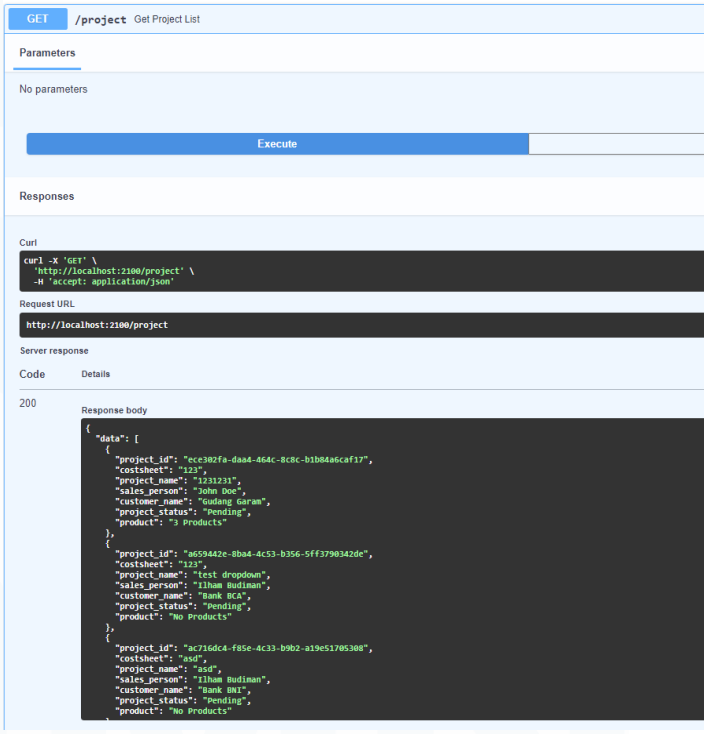
Gambar 3. 42 Bagian *backend code*

Pada Gambar 3.42 merupakan contoh dari beberapa API yang digunakan dengan *tag contract*. *Tags* disini sangat bermanfaat untuk mengatur API yang ada dengan mengkategorikan API berdasarkan fungsinya. Sehingga pada dokumen akan lebih terstruktur.



Gambar 3. 43 *Docs* API *contract*

Pada Gambar 3.43 merupakan contoh dari *Document* API yang dibuat oleh mahasiswa, contoh menggambarkan *enpoint* dengan kategori *Contract*. *Document* API ini penting bagi *developer* dikarenakan pada *document* ini *developer* dapat melakukan *test* terhadap API yang akan digunakan, sehingga mengurang terjadinya *error* disaat API akan di-*consume* oleh *frontend application*.



Gambar 3. 44 Contoh penggunaan *document*

Pada Gambar 3.44 merupakan contoh testing dari *endpoint* yang digunakan dalam aplikasi, contoh mengembalikan data data list proyek yang disimpan dalam *database*. Dengan begitu nantinya dapat dilihat *response* yang dikeluarkan saat melakukan *test* terhadap API yang digunakan tanpa perlu *tools* tambahan.

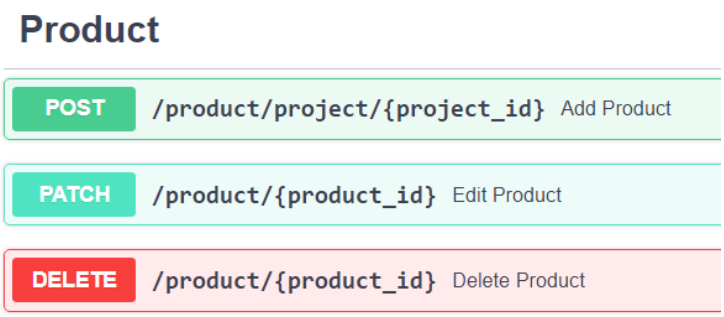
- Product

  Sama seperti *service contract*, *service product* merupakan *service* yang berfungsi untuk meng-*handle* segala proses yang berkaitan dengan *product*.

```python
from fastapi import APIRouter, Header    Import "fastapi" could not be resolved
from . import schema, service as ProductService
from typing import Annotated, Optional

product_router = APIRouter()

@product_router.post('/product/project/{project_id}', tags=['Product'])
async def add_product(
    request: schema.Product,
    project_id: str,
    pm_id: Annotated[Optional[str], Header()] = None,
    cm_id: Annotated[Optional[str], Header()] = None,
    sla_id: Annotated[Optional[str], Header()] = None,
    implementation_id: Annotated[Optional[str], Header()] = None
    ):
    add_product_response = ProductService.add_product(request, project_id,
                                                      pm_id, cm_id, sla_id,
                                                      implementation_id)
    return add_product_response

@product_router.patch('/product/{product_id}', tags=['Product'])
async def edit_product(request: schema.Product, product_id: str):
    edit_product_response = ProductService.edit_product(request, product_id)
    return edit_product_response

@product_router.delete('/product/{product_id}', tags=['Product'])
async def delete_product(product_id: str):
    delete_product_response = ProductService.delete_product(product_id)
    return delete_product_response
```

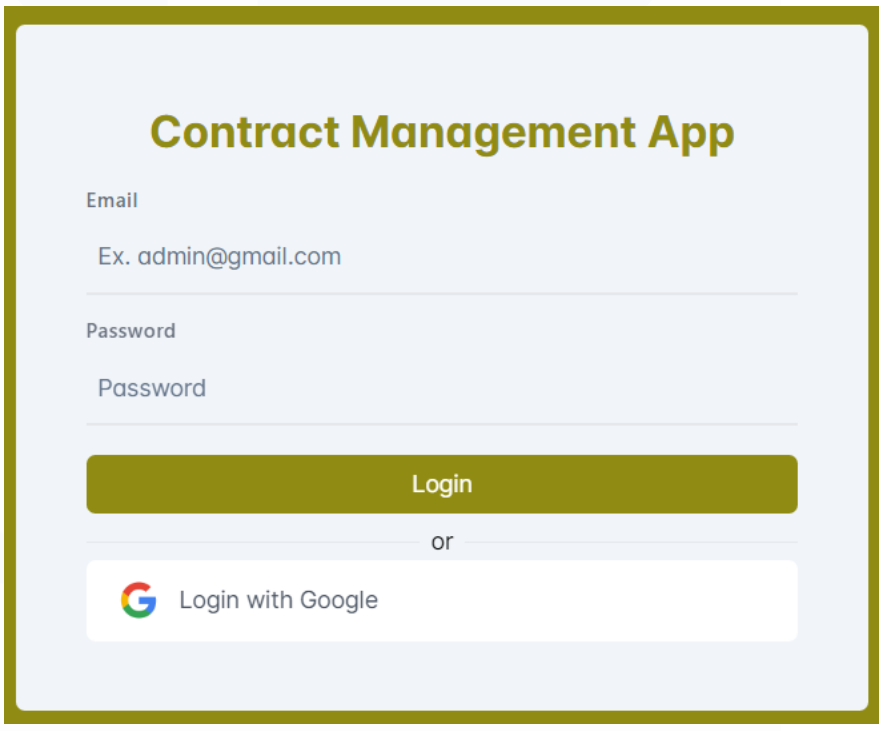
Gambar 3. 45 *Product Service Code*

Pada Gambar 3.45. merupakan sebuah barisan kode dalam sebuah *endpoint* yang digunakan dalam aplikasi yang didevelop. Untuk *tag product* digunakan untuk meng-kategorikan *list* API *product* sehingga dapat mudah untuk dilakukan *testing* sebelum dilakukan *request* dalam *frontend*.



Gambar 3. 46 *Docs* API *product*

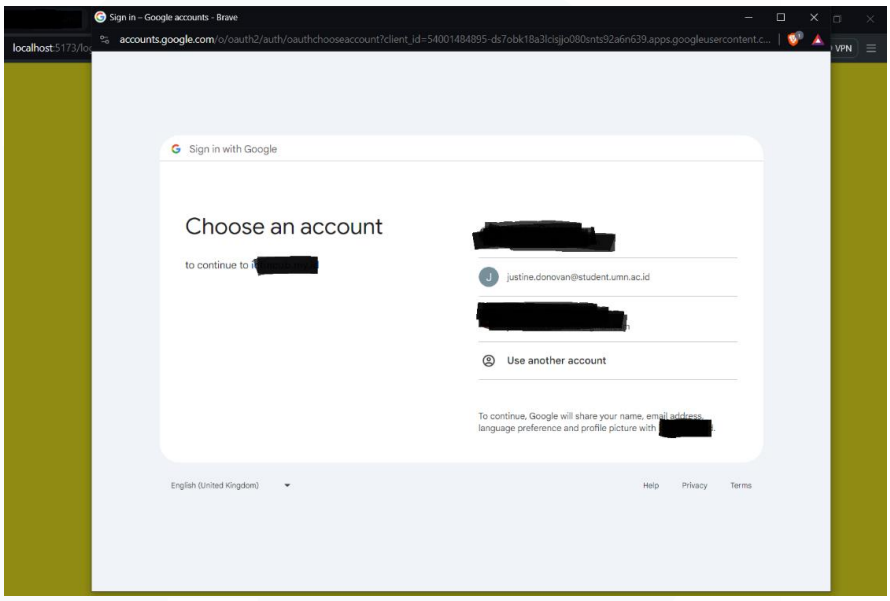
Seperti yang sudah diperlihatkan gambar gambar contohny dalam Gambae 3.46 merupakan bagian dari *service* penting yang berjalan pada aplikasi *Contract Management* selanjutnya mahasiswa membuat tampilan *web* menggunakan VUE, mahasiswa juga melakukan *manage* user contohnya membuat Login Page.



Gambar 3. 47 Login Form

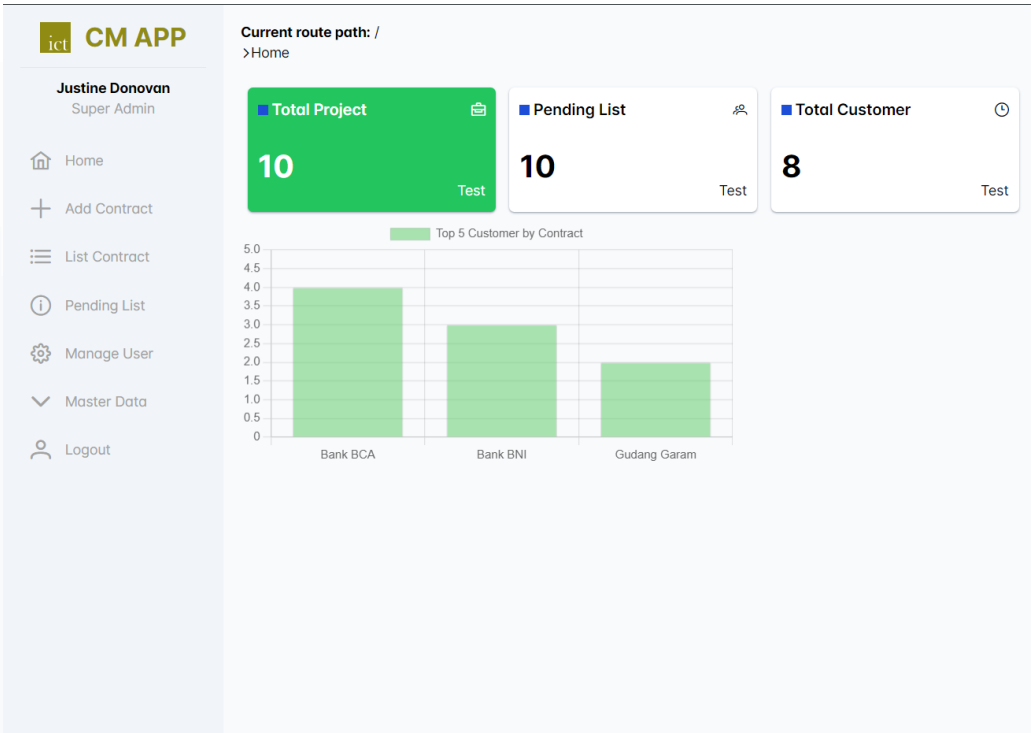
Pada Gambar 3.47 merupakan *form login* yang dibuat oleh mahasiswa yang telah terhubung dengan akun Google sehingga

dapat mempermudah *user* jika ingin melakukan *login* pada aplikasi *contract management*.



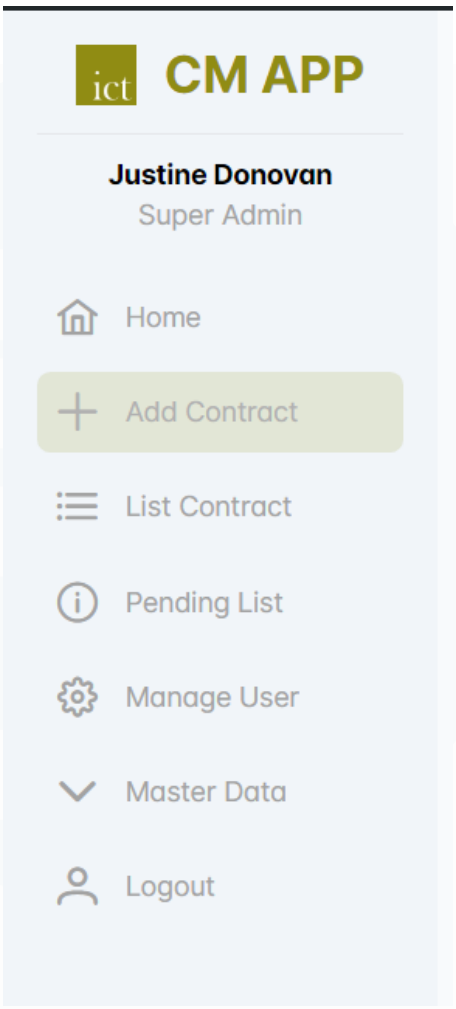
Gambar 3. 48 Login dengan Google

Pada Gambar 3.48. menunjukan menu pilihan email yang dimiliki oleh user untuk melakukan Login ke aplikasi. Login dengan Google ini dapat dibuat tanpa masalah dengan menggunakan PocketBase. Selain itu terdapat tampilan *dashboard* yang dibuat oleh mahasiswa dengan bantuan *library* Chart.js, fungsi halaman *dashboard* ini digunakan untuk menujukan data-data yang penting.



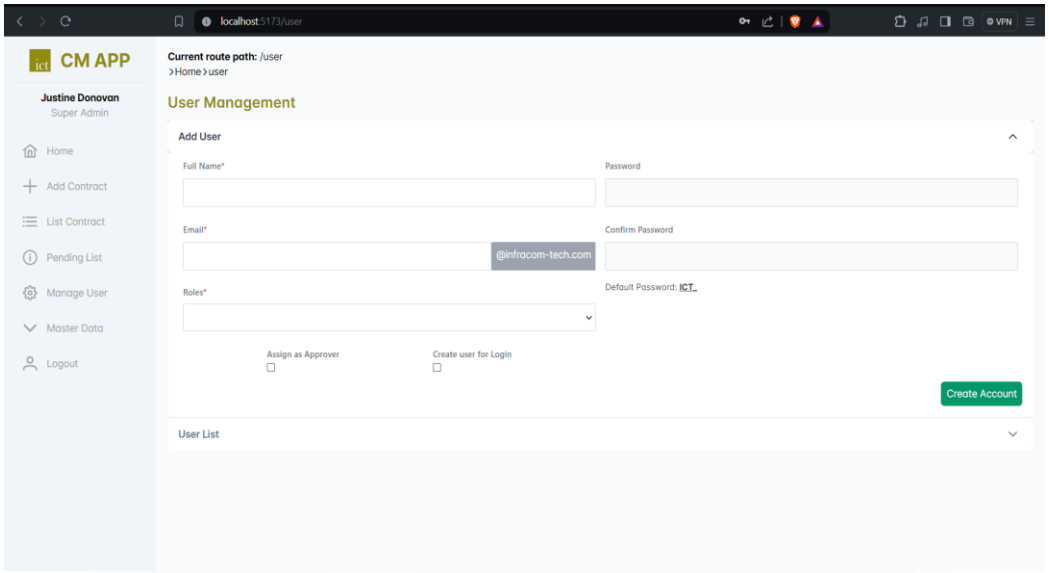
Gambar 3. 49 Tampilan Dashboard Aplikasi

Pada Gambar 3.49. menunjukan data data yang ditampilkan merupakan data manipulasi. Dengan adanya *dashboard* pada sebuah aplikasi tentunya akan mempermudah dalam mendapatkan informasi yang dibutuhkan sehingga *user* dapat melihat progress yang dimiliki oleh perusahaan. Untuk mudah melakukan navigasi pada aplikasi, mahasiswa membuat *component sidebar* yang dapat digunakan oleh user untuk melakukan navigasi antar halaman pada aplikasi.



Gambar 3. 50 Sidebar *application*

Gambar 3.50. merupakan contoh dari komponen navigation yang dibuat dalam bentuk komponen sehingga semua halaman yang terkait dalam aplikasi dapat menggunakan komponsen tersebut tanpa harus menulis ulang kode. Dan beberapa halaman dibawah ini merupakan halaman yang dibuat oleh mahasiswa dalam kontribusi yang dilakukan oleh mahasiswa dalam proyek *Contract Management*.



Gambar 3. 51 Tampilan User Management

Pada Gambar 3.51. berfungsi untuk sebuah *admin* untuk mendaftarkan pengguna baru dalam aplikasi, sehingga tidak sembarang pengguna dapat mengakses *web*. Pada proses ini dibuat dengan melibatkan beberapa *service* dikarenakan untuk membuat halaman *User Management* ini dibutuhkan 2 (dua) *service* yang terlibat didalamnya.

3.3 Kendala yang Ditemukan

Dalam proses berjalannya kerja magang, mahasiswa memiliki beberapa kendala yang dihadapi. Diantaranya yaitu:

1. Menemukan masalah teknis yang rumit. Masalah ini biasanya ditemukan dimana terjadi sebuah bug pada aplikasi atau sebuah kondisi dimana aplikasi tidak berjalan sebagaimana mestinya.
2. Menentukan design aplikasi, untuk menentukan design akhir dari aplikasi yang akan dibangun tentunya menjadi sebuah tantangan yang dapat dihadapi oleh dikarenakan perubahan yang terjadi merupakan hal yang wajar, dikarenakan pada saat *client* merasa beberapa fitur dalam aplikasi yang harus ditingkatkan.

3. Beradaptasi dari sebuah *tools*/*framework* ke *tools*/*framework* yang lainnya, mahasiswa harus mampu beradaptasi secara cepat dikarenakan teknologi yang digunakan pada aplikasi tentunya tidak akan selalu sama berdasarkan tujuan ataupun kehendak dari *client*.
4. Keterbatasan informasi internal, maksud dari permasalahan ini dikarenakan mahasiswa berstatus sebagai *Intern* pada perusahaan terkait maka dari itu sesuai kebijakan perusahaan mahasiswa memiliki keterbatasan mengakses infromasi internal demi menghargai infromasi perusahaan dan juga *client*.

3.4 Solusi atas Kendala yang Ditemukan

Berdasarkan permasalahan permasalahan yang dihadapi mahasiswa, berikut beberapa hal yang dapat dilakukan untuk mengatasi masalah masalah :

1. Mempersiapkan diri sebelum proyek berjalan sehingga saat proyek berjalan kendala yang dialami tidak terlalu menghambat waktu mahasiswa dalam mengerjakan sebuah fitur atau proses dalam aplikasi yang ingin dibuat.
2. Melakukan *research* mengenai desain UI/UX dan juga mencari referensi desain yang memungkinkan untuk memunculkan ide dalam mendesain aplikasi.
3. Membaca dokumentasi yang terkait dengan *tools/framework* yang akan digunakan, sehingga dapat lebih menambah pengetahuan bahwa fungsi apa saja yang dapat digunakan pada *tools/framework* tersebut.
4. Memperhatikan informasi yang diberikan oleh anggota tim sehingga mahasiswa mendapatkan informasi yang penting dalam berjalannya sebuah proyek.