LAMPIRAN

lampiran-1 loginActivity

```
import { SignInForm } from
"@/components/auth/form/sign-
in-form";

export default function Page()
{
  return (
    <div className="flex min-
h-svh w-full items-center
justify-center p-6 md:p-10">
      <div className="w-full
max-w-sm mx-auto my-auto">
        <SignInForm />
      </div>
    </div>
  );
}
```

Lampiran-2 logoutActivity

```
import { SignUpForm } from
"@/components/auth/form/sign-
up-form";

export default function Page()
{
  return (
    <div className="flex min-
h-svh w-full items-center
justify-center p-6 md:p-10">
      <div className="w-full
max-w-sm">
        <SignUpForm />
      </div>
    </div>
  );
}
```

Lampiran-3 MAainActivity

```
"use client";
import { Dropzone,
DropzoneContent,
DropzoneEmptyState } from
"@/components/dropzone";
import { FieldInfo } from
"@/components/field-info";
import { NavBreadcrumb } from
"@/components/nav-breadcrumb";
import { Button } from
"@/components/ui/button";
import { Label } from
"@/components/ui/label";
import { Select,
SelectContent, SelectItem,
```

```
SelectTrigger, SelectValue }
from "@/components/ui/select";
import { useCreateBanner }
from "@/hooks/use-banner";
import { useCourse } from
"@/hooks/use-course";
import { useSupabaseUpload }
from "@/hooks/use-supabase-
upload";
import { bannerFormSchema,
TBannerForm } from
"@/schema/banner-schema";
import { Course } from
"@prisma/client";
import { useForm } from
"@tanstack/react-form";
import { ArrowLeftIcon,
Loader2 } from "lucide-react";
import { useRouter } from
"next/navigation";
import { toast } from
"sonner";

export default function Page()
{
  const router = useRouter();
  const { getAll } =
useCourse({ enableGet: true
});
  const { mutate, isPending }
= useCreateBanner();

  const imageUploadProps =
useSupabaseUpload({
    bucketName:
process.env.NEXT_PUBLIC_SUPABA
SE_BUCKET!,
    path: "test/images",
    allowedMimeTypes:
["image/*"],
    maxFiles: 1,
    maxFileSize: 1000 * 1000 *
20,
  });

  const form = useForm({
    defaultValues: {
      image: "",
      courseId: "",
    } as TBannerForm,
    validators: { onSubmit:
bannerFormSchema },
    onSubmit: async (values)
=> {
```

```
      console.log("Form data
before submission:", values);
// Debugging form data

      // Upload files to
Supabase
      const { success:
imageSuccess, error:
imageError } = await
imageUploadProps.onUpload();

      if
(!imageSuccess.length) {
        toast.error("Silakan
pilih gambar");
        return;
      }

      // Handle file upload
errors
      if (imageError.length >
0) {

toast.warning(`Sebagian file
gagal diupload:
${[...imageError].map((e) =>
e.name).join(", ")}`);
      }

      const uploadedImagePaths
= imageSuccess.map((filename)
=> `test/images/${filename}`);

      mutate({
        ...values.value,
        image:
uploadedImagePaths[0],
      });
    },
  });
  return (
    <form
      onSubmit={(e) => {
        e.preventDefault();
        e.stopPropagation();
        form.handleSubmit();
      }}
      className="space-y-8"
    >
      <NavBreadcrumb
current="/beranda/banner/creat
e" />
      <section className="flex
items-center gap-3">
        <Button size="icon"
variant="outline"
```

```
className="rounded-full"
onClick={() => router.back()}>
          <ArrowLeftIcon
size={18} />
        </Button>
        <h2 className="text-
2xl font-bold tracking-
tight">Buat Banner</h2>
      </section>

      <div className="space-y-
3">
        <Label>Gambar</Label>
        <Dropzone
{...imageUploadProps}>
          <DropzoneEmptyState
/>
          <DropzoneContent />
        </Dropzone>
      </div>

      <form.Field
name="courseId">
        {(field) => (
          <div
className="space-y-2">
            <Label
htmlFor={field.name}>Pendidika
n Terakhir</Label>
            <Select
value={field.state.value}
onValueChange={(val) =>
field.handleChange(val)}>
              <SelectTrigger
id={field.name} className="w-
full">
                <SelectValue
placeholder="Pilih Kursus" />
              </SelectTrigger>
              <SelectContent>

{getAll.data?.map((course:
Course) => (
                  <SelectItem
key={course.id}
value={course.id}>

{course.name}

</SelectItem>
                ))}
              </SelectContent>
            </Select>
```

```
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      <div className="pt-4">
        <form.Subscribe
selector={(s) =>
[s.isSubmitting]}>
          {([isSubmitting]) =>
(
            <Button
type="submit" className="w-
full" disabled={isPending ||
isSubmitting}>
              {(isPending ||
isSubmitting) && <Loader2
className="mr-2 h-4 w-4
animate-spin" />}
              Simpan
            </Button>
          )}
        </form.Subscribe>
      </div>
    </form>
  );
}
```

```
"use client";

import { Empty } from
"@/components/empty";
import ListRenderer from
"@/components/list-renderer";
import { NavBreadcrumb } from
"@/components/nav-breadcrumb";
import { ProgramItem } from
"@/components/program/item";
import { Button } from
"@/components/ui/button";
import { useCourse } from
"@/hooks/use-course";
import { Course, CourseProgram
} from "@prisma/client";
import { ArrowLeft } from
"lucide-react";
import { useRouter } from
"next/navigation";
import { use, useMemo,
useState } from "react";

export default function Page({
params }: { params: Promise<{
courseId: string }> }) {
  const { back } =
useRouter();
```

```
  const { courseId } =
use(params);

  const { getById } =
useCourse({ enableGet: true,
id: courseId });
  const [activeFilters,
setActiveFilters] =
useState<string[]>([]);

  // generate category counts
  const categoryCounts =
useMemo(() => {
    const counts:
Record<string, number> = {};
    if
(getById.data?.courseProgram)
{

getById.data.courseProgram.for
Each((p: CourseProgram) => {
        counts[p.category] =
(counts[p.category] || 0) + 1;
      });
    }
    return counts;
  },
[getById.data?.courseProgram])
;

  // unique categories
(prevent duplicates)
  const uniqueCategories =
useMemo(() => {
    if
(!getById.data?.courseProgram)
return [];
    const seen = new
Set<string>();
    return
getById.data.courseProgram.fil
ter((p: CourseProgram) => {
      if
(seen.has(p.category)) return
false;
      seen.add(p.category);
      return true;
    });
  },
[getById.data?.courseProgram])
;

  // toggle filter
```

```
  const toggleFilter =
(category: string) => {
    if (category === "ALL") {
      setActiveFilters([]); //
reset filters
    } else {
      setActiveFilters((prev)
=> (prev.includes(category) ?
prev.filter((f) => f !==
category) : [...prev,
category]));
    }
  };

  // filtered program list
  const filteredPrograms =
    activeFilters.length > 0
&& getById.data?.courseProgram
      ?
getById.data.courseProgram.fil
ter((p: CourseProgram) =>
activeFilters.includes(p.categ
ory))
      :
getById.data?.courseProgram;

  return (
    <main className="space-y-
8">
      {/* Breadcrumb
navigation */}
      <NavBreadcrumb
current="/beranda/course/detai
l" />

      {/* Header with back
button and course name */}
      <section className="flex
gap-2 items-center">
        <Button size="icon"
variant="outline"
className="rounded-full"
onClick={() => back()}>
          <ArrowLeft />
        </Button>
        <h1 className="text-xl
sm:text-2xl font-
bold">{getById.data?.name}</h1
>
      </section>

      {/* Filter buttons */}
      <div className="flex
gap-2 overflow-auto pb-5">
```

```
        {/* "All" button with
total count */}
        <Button
          size="sm"
          className="rounded-
full"

variant={activeFilters.length
=== 0 ? "default" :
"secondary"}
          onClick={() =>
toggleFilter("ALL")}
        >
          All
({getById.data?.courseProgram?
.length || 0})
        </Button>

        {/* Category-specific
buttons */}

{uniqueCategories.map((program
: CourseProgram) => (
          <Button
            size="sm"
            key={program.id}

className="rounded-full"

variant={activeFilters.include
s(program.category) ?
"default" : "secondary"}
            onClick={() =>
toggleFilter(program.category)
}
          >
            {program.category}
({categoryCounts[program.categ
ory] || 0})
          </Button>
        ))}
      </div>

      {/* Render program list
*/}

<ListRenderer<CourseProgram &
{ Course: Course }>
        of={filteredPrograms}
        render={(items) =>
<ProgramItem program={items}
/>}
        className="grid grid-
cols-1 sm:grid-cols-2 lg:grid-
cols-3 gap-4"
```

```
loadingState={<h1>Loading...</
h1>}

isLoading={getById.isLoading}
        emptyState={<Empty
text="No course program found"
/>}
      />
    </main>
  );
}
"use client";
import { NavBreadcrumb } from
"@/components/nav-breadcrumb";
import { columns } from
"@/components/order/table/colu
mns";
import { DataTable } from
"@/components/order/table/data
-table";
import { Input } from
"@/components/ui/input";
import {
useGetOrderByUserAdmin } from
"@/hooks/use-order";
import { useRouter,
useSearchParams } from
"next/navigation";
import { useDebouncedCallback
} from "use-debounce";

export default function Page()
{
  const params =
useSearchParams();
  const router = useRouter();
  const orderId =
params.get("orderId") || "";
  const { data } =
useGetOrderByUserAdmin({
orderId });

  // gunakan
useDebouncedCallback agar bisa
dipanggil langsung
  const handleSearch =
useDebouncedCallback((value:
string) => {
    router.replace(`/list-
order?orderId=${value}`);
  }, 500);

  return (
```

```
    <main className="space-y-
8">
      {/* Breadcrumb */}
      <NavBreadcrumb
current="/beranda/list-order"
/>

      {/* Header Section */}
      <section className="flex
flex-col md:flex-row gap-5
justify-between items-start
md:items-center">
        <h1 className="text-xl
sm:text-2xl font-bold">Orderan
Kursus</h1>
        <Input
          className="w-full
md:w-72 rounded-full"
          placeholder="Cari
orderan..."

defaultValue={orderId}
          onChange={(e) =>
handleSearch(e.target.value)}
        />
      </section>

      <DataTable
columns={columns} data={data
|| []} />
    </main>
  );
}
"use client";
import { Empty } from
"@/components/empty";
import ListRenderer from
"@/components/list-renderer";
import { MemberItems } from
"@/components/member/item";
import { NavBreadcrumb } from
"@/components/nav-breadcrumb";
import { OrderWithRelations }
from
"@/components/order/item";
import { useGetOrderByMember }
from "@/hooks/use-order";
export default function Page()
{
  const { data, isLoading } =
useGetOrderByMember();
  console.log("Order Data:",
data);
  return (
```

```
    <main className="space-y-
8">
      {/* Breadcrumb */}
      <NavBreadcrumb
current="/beranda/member" />
      {/* Header Section */}
      <section className="flex
flex-col md:flex-row gap-5
justify-between items-start
md:items-center">
        <h1 className="text-xl
sm:text-2xl font-bold">Daftar
Member</h1>
      </section>

<ListRenderer<OrderWithRelatio
ns>
        of={data}
        render={(items) =>
<MemberItems item={items} />}
        className="grid grid-
cols-1 sm:grid-cols-2 lg:grid-
cols-2 gap-4"

loadingState={<h1>Loading...</
h1>}
        isLoading={isLoading}
        emptyState={<Empty
text="Tidak ada member" />}
      />
    </main>
  );
}
"use client";
import { FieldInfo } from
"@/components/field-info";
import { NavBreadcrumb } from
"@/components/nav-breadcrumb";
import { Button } from
"@/components/ui/button";
import { Card } from
"@/components/ui/card";
import { Input } from
"@/components/ui/input";
import { Label } from
"@/components/ui/label";
import { Textarea } from
"@/components/ui/textarea";
import { useProgram } from
"@/hooks/use-program";
import { programFromSchema,
TProgramForm } from
"@/schema/program-schema";
import { useForm } from
"@tanstack/react-form";
```

```
import { ArrowLeftIcon,
Loader2 } from "lucide-react";
import { useRouter } from
"next/navigation";
import { use } from "react";

const DAYS_OF_WEEK = [
  { id: "senin", label:
"Senin" },
  { id: "selasa", label:
"Selasa" },
  { id: "rabu", label: "Rabu"
},
  { id: "kamis", label:
"Kamis" },
  { id: "jumat", label:
"Jumat" },
  { id: "sabtu", label:
"Sabtu" },
];

export default function Page({
params }: { params: Promise<{
courseId: string }> }) {
  const { courseId } =
use(params);
  const { create } =
useProgram({});
  const { back } =
useRouter();

  const form = useForm({
    defaultValues: {
      name: "",
      price: "",
      description: "",
      category: "",
      sessions: "",
      benefits: [""] as
string[],
      schedule: [] as
string[],
      mentorName: "",
      materials: [""] as
string[],
    } as TProgramForm,
    validators: { onSubmit:
programFromSchema },
    onSubmit: async (values)
=> {

console.log(values.value);
      create.mutate({
        ...values.value,
        courseId,
      });
```

```
    },
  });

  return (
    <form
      onSubmit={(e) => {
        e.preventDefault();
        e.stopPropagation();
        form.handleSubmit();
      }}
      className="space-y-8"
    >
      <NavBreadcrumb
current="/beranda/my-
course/program/create" />
      <section className="flex
items-center gap-3">
        <Button size="icon"
variant="outline"
className="rounded-full"
onClick={back}>
          <ArrowLeftIcon
size={18} />
        </Button>
        <h2 className="text-
2xl font-bold tracking-
tight">Buat Program</h2>
      </section>

        {/* Name */}
        <form.Field name="name">
          {(field) => (
            <div
className="space-y-3">
              <Label
htmlFor={field.name}>Nama
Program</Label>
              <Input

placeholder="Tulis nama
program pelatihan..."
                id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
                onChange={(e) =>
field.handleChange(e.target.va
lue)}
              />
              <FieldInfo
field={field} />
            </div>
          )}
```

```
      </form.Field>

      {/* Category */}
      <form.Field
name="category">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Category
Program</Label>
            <Input

placeholder="Tulis category
program..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* sessions */}
      <form.Field
name="sessions">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Jumlah
Pertemuan</Label>
            <Input

placeholder="Tulis jumlah
pertemuan program..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
```

```
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Mentor Name */}
      <form.Field
name="mentorName">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Nama
Mentor</Label>
            <Input

placeholder="Tulis nama mentor
program..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
                onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Schedule */}
      <form.Field
name="schedule">
        {(field) => (
          <div
className="space-y-3">
            <Label>Jadwal
Program</Label>
            <div
className="grid grid-cols-2
gap-3 sm:grid-cols-3 lg:grid-
cols-4">

{DAYS_OF_WEEK.map((day) => {
                const
isSelected =
field.state.value.includes(day
.id);
```

```
                return (
                  <Card
key={day.id}
                    
className={`p-3 cursor-pointer
shadow-none ${
                      
isSelected ? "bg-primary/10
border-primary" : "border-
border"
                    }`}
                    
onClick={() => {
                      if
(isSelected) {
                        
field.handleChange(field.state
.value.filter((d) => d !==
day.id));
                      } else {
                        
field.handleChange([...field.s
tate.value, day.id]);
                      }
                    }}
                  >
                    <div
className="text-center">
                      <div
className="text-sm font-
medium">{day.label}</div>
                    </div>
                  </Card>
                );
              })}
            </div>

{field.state.value.length > 0
&& (
              <div
className="text-sm text-muted-
foreground">
                Jadwal
terpilih:{" "}
                
{field.state.value.map((id) =>
DAYS_OF_WEEK.find((day) =>
day.id === id)?.label).join(",
")}
              </div>
            )}
            <FieldInfo
field={field} />
          </div>
```

```
        )}
      </form.Field>

      {/* Price */}
      <form.Field
name="price">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Harga
(Rp)</Label>
            <Input
              type="number"

placeholder="100000"
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Description */}
      <form.Field
name="description">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Deskripsi
Program</Label>
            <Textarea

placeholder="Tulis deskripsi
lengkap program ini..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
```

```
                onChange={(e) =>
field.handleChange(e.target.va
lue)}
              />
              <FieldInfo
field={field} />
            </div>
          )}
        </form.Field>

        {/* Benefits */}
        <form.Field
name="benefits">
          {(field) => (
            <div
className="space-y-3">
              <Label>Manfaat
Program</Label>

{field.state.value.map((benefi
t, index) => (
                <div key={index}
className="flex gap-2 items-
center">
                  <Input

placeholder={`Manfaat ke-
${index + 1}`}
value={benefit}

onChange={(e) => {
                      const
updated =
[...field.state.value];

updated[index] =
e.target.value;

field.handleChange(updated);
                    }}
                  />
                  <Button

type="button"

variant="outline"
                    size="icon"
                    onClick={()
=>
field.handleChange(field.state
.value.filter((_, i) => i !==
index))}
                  >
                    X
                  </Button>
```

```
            </div>
          ))}
          <Button
            type="button"
            size="sm"
variant="outline"
            onClick={() =>
field.handleChange([...field.s
tate.value, ""])}
          >
            Tambah Manfaat
          </Button>
          <FieldInfo
field={field} />
        </div>
      )}
    </form.Field>

    {/* Materials */}
    <form.Field
name="materials">
      {(field) => (
        <div
className="space-y-3">
          <Label>Materi
Program</Label>
{field.state.value.map((materi
al, index) => (
            <div key={index}
className="flex gap-2 items-
center">
              <Input
placeholder={`Materi ke-
${index + 1}`}
value={material}
onChange={(e) => {
                const
updated =
[...field.state.value];
updated[index] =
e.target.value;
field.handleChange(updated);
              }}
            />
            <Button
type="button"
```

```
                                        </form.Field>

variant="outline"
                  size="icon"           <Button
                  onClick={()           disabled={create.isPending}
=>                                      className="w-full"
field.handleChange(field.state          type="submit">
.value.filter((_, i) => i !==             {create.isPending &&
index))}                                <Loader2 className="mr-2 h-4
                >                       w-4 animate-spin" />}
                  X                      Simpan Program
                  </Button>             </Button>
                </div>                 </form>
              ))}                     );
              <Button                  }
                type="button"          "use client";
                size="sm"              import { FieldInfo } from
                                       "@/components/field-info";
variant="outline"                      import { NavBreadcrumb } from
                onClick={() =>         "@/components/nav-breadcrumb";
field.handleChange([...field.s         import { Button } from
tate.value, ""])}                      "@/components/ui/button";
              >                        import { Card } from
                Tambah Materi          "@/components/ui/card";
              </Button>                import { Input } from
              <FieldInfo               "@/components/ui/input";
field={field} />                       import { Label } from
            </div>                     "@/components/ui/label";
          )}
```

```
import { Textarea } from
"@/components/ui/textarea";
import { useProgram } from
"@/hooks/use-program";
import { programFromSchema,
TProgramForm } from
"@/schema/program-schema";
import { useForm } from
"@tanstack/react-form";
import { ArrowLeftIcon,
Loader2 } from "lucide-react";
import { useRouter } from
"next/navigation";
import { use } from "react";

const DAYS_OF_WEEK = [
  { id: "senin", label:
"Senin" },
  { id: "selasa", label:
"Selasa" },
  { id: "rabu", label: "Rabu"
},
  { id: "kamis", label:
"Kamis" },
  { id: "jumat", label:
"Jumat" },
  { id: "sabtu", label:
"Sabtu" },
];

export default function Page({
params }: { params: Promise<{
programId: string }> }) {
  const { programId } =
use(params);
  const { update, getById,
deleteProgram } = useProgram({
enableGet: true, id: programId
});
  const { back } =
useRouter();

  const form = useForm({
    defaultValues: {
      name: getById.data?.name
|| "",
      price:
String(getById.data?.price) ||
"",
      description:
getById.data?.description ||
"",
      category:
String(getById.data?.category)
|| "",
```

```
      benefits:
getById.data?.benefits || [],
      materials:
getById.data?.materials || [],
      schedule:
getById.data?.schedule || [],
      mentorName:
getById.data?.mentorName ||
"",
      sessions:
String(getById.data?.sessions)
|| "",
    } as TProgramForm,
    validators: { onSubmit:
programFromSchema },
    onSubmit: async (values)
=> {

console.log(values.value);
      update.mutate({
        ...values.value,
        courseId: programId,
      });
    },
  });

  const handleDelete = async
() => {
deleteProgram.mutate(programId
);
  };

  return (
    <form
      onSubmit={(e) => {
        e.preventDefault();
        e.stopPropagation();
        form.handleSubmit();
      }}
      className="space-y-6"
    >
      <NavBreadcrumb
current="/beranda/my-
course/program/update" />
      <section className="flex
items-center gap-3">
        <Button type="button"
size="icon" className="w-7 h-
7" onClick={back}>
          <ArrowLeftIcon />
        </Button>
        <h2 className="text-
2xl font-bold tracking-
tight">Update Program</h2>
      </section>
```

```
      {/* Name */}
      <form.Field name="name">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Nama
Program</Label>
            <Input
placeholder="Tulis nama
program pelatihan..."
              id={field.name}
name={field.name}
value={field.state.value}
onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Category */}
      <form.Field
name="category">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Category
Program</Label>
            <Input
placeholder="Tulis category
program..."
              id={field.name}
name={field.name}
value={field.state.value}
onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
```

```
          </div>
        )}
      </form.Field>

      {/* sessions */}
      <form.Field
name="sessions">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Jumlah
Pertemuan</Label>
            <Input

placeholder="Tulis jumlah
pertemuan program..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Mentor Name */}
      <form.Field
name="mentorName">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Nama
Mentor</Label>
            <Input

placeholder="Tulis nama mentor
program..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
```

```
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Schedule */}
      <form.Field
name="schedule">
        {(field) => (
          <div
className="space-y-3">
            <Label>Jadwal
Program</Label>
            <div
className="grid grid-cols-2
gap-3 sm:grid-cols-3 lg:grid-
cols-4">

{DAYS_OF_WEEK.map((day) => {
              const
isSelected =
field.state.value.includes(day
.id);
              return (
                <Card
key={day.id}

className={`p-3 cursor-pointer
shadow-none ${

isSelected ? "bg-primary/10
border-primary" : "border-
border"
                  }`}

onClick={() => {
                    if
(isSelected) {

field.handleChange(field.state
.value.filter((d) => d !==
day.id));
                    } else {

field.handleChange([...field.s
tate.value, day.id]);
                    }
                  }}
                >
```

```
                    <div
className="text-center">
                      <div
className="text-sm font-
medium">{day.label}</div>
                    </div>
                  </Card>
                );
              })}
            </div>

{field.state.value.length > 0
&& (
              <div
className="text-sm text-muted-
foreground">
                Jadwal
terpilih:{" "}

{field.state.value.map((id) =>
DAYS_OF_WEEK.find((day) =>
day.id === id)?.label).join(",
")}
              </div>
            )}
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Price */}
      <form.Field
name="price">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Harga
(Rp)</Label>
            <Input

placeholder="100000"
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}

            />
            <FieldInfo
field={field} />
```

```
          </div>
        )}
      </form.Field>

      {/* Description */}
      <form.Field
name="description">
        {(field) => (
          <div
className="space-y-3">
            <Label
htmlFor={field.name}>Deskripsi
Program</Label>
            <Textarea

placeholder="Tulis deskripsi
lengkap program ini..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Benefits */}
      <form.Field
name="benefits">
        {(field) => (
          <div
className="space-y-3">
            <Label>Manfaat
Program</Label>

{field.state.value.map((benefi
t, index) => (
              <div key={index}
className="flex gap-2 items-
center">
                <Input

placeholder={`Manfaat ke-
${index + 1}`}

value={benefit}

onChange={(e) => {
```

```
                    const
updated =
[...field.state.value];

updated[index] =
e.target.value;

field.handleChange(updated);
                  }}
                />
                <Button

type="button"

variant="outline"
                  size="icon"
                  onClick={()
=>
field.handleChange(field.state
.value.filter((_, i) => i !==
index))}
                >
                  X
                </Button>
              </div>
            ))}
            <Button
              type="button"
                  size="sm"

variant="outline"
                  onClick={() =>
field.handleChange([...field.s
tate.value, ""])}
                >
                  Tambah Manfaat
                </Button>
                <FieldInfo
field={field} />
              </div>
            )}
          </form.Field>

          {/* Materials */}
          <form.Field
name="materials">
            {(field) => (
              <div
className="space-y-3">
                <Label>Materi
Program</Label>

{field.state.value.map((materi
al, index) => (
```

```
                <div key={index}
className="flex gap-2 items-
center">
                  <Input
placeholder={`Materi ke-
${index + 1}`}
value={material}
onChange={(e) => {
                      const
updated =
[...field.state.value];
updated[index] =
e.target.value;
field.handleChange(updated);
                    }}
                  />
                  <Button
type="button"
variant="outline"
                    size="icon"
                    onClick={()
=>
field.handleChange(field.state
.value.filter((_, i) => i !==
index))}
                  >
                    X
                  </Button>
                </div>
              ))}
              <Button
                type="button"
                size="sm"
variant="outline"
                onClick={() =>
field.handleChange([...field.s
tate.value, ""])}
              >
                Tambah Materi
              </Button>
              <FieldInfo
field={field} />
            </div>
          )}
        </form.Field>
        <div className="grid
grid-cols-2 gap-3">
```

```
        <Button
disabled={deleteProgram.isPend
ing}
          className="w-full"
          type="button"
variant={"destructive"}
onClick={handleDelete}
        >
{deleteProgram.isPending &&
<Loader2 className="mr-2 h-4
w-4 animate-spin" />}
          Delete Program
        </Button>
        <Button
disabled={update.isPending}
className="w-full"
type="submit">
          {update.isPending &&
<Loader2 className="mr-2 h-4
w-4 animate-spin" />}
          Update Program
        </Button>
      </div>
    </form>
  );
}
"use client";
import { Empty } from
"@/components/empty";
import ListRenderer from
"@/components/list-renderer";
import { NavBreadcrumb } from
"@/components/nav-breadcrumb";
import { ProgramItem } from
"@/components/program/item";
import { Button } from
"@/components/ui/button";
import { useCourse } from
"@/hooks/use-course";
import { Course, CourseProgram
} from "@prisma/client";
import { ArrowLeftIcon, Plus }
from "lucide-react";
import Link from "next/link";
import { useRouter } from
"next/navigation";
import { use } from "react";
export default function Page({
params }: { params: Promise<{
courseId: string }> }) {
  const { courseId } =
use(params);
```

```
  const { back } =
useRouter();
  const { getById } =
useCourse({ enableGet: true,
id: courseId });

  return (
    <main className="space-y-
8">
      <NavBreadcrumb
current="/beranda/my-
course/detail" />
      <section className="flex
items-end justify-between">
        <div className="flex
items-center gap-2">
          <Button size="icon"
variant="outline"
className="rounded-full"
onClick={back}>
            <ArrowLeftIcon
size={18} />
          </Button>
          <h2 className="text-
2xl font-bold tracking-
tight">{getById.data?.name}</h
2>
        </div>
        <div className="flex
gap-3">
          <Button asChild>
            <Link href={`/my-
course/${courseId}/create`}>
              <Plus />
              Buat Program
            </Link>
          </Button>
        </div>
      </section>

<ListRenderer<CourseProgram &
{ Course: Course }>

of={getById?.data?.courseProgr
am}
        render={(items) =>
<ProgramItem program={items}
/>}
        className="grid grid-
cols-1 sm:grid-cols-2 lg:grid-
cols-3 gap-4"

loadingState={<h1>Loading...</
h1>}
```

```
isLoading={getById.isLoading}
        emptyState={<Empty
text="Tidak ada program yang
ditemukan" />}
      />
    </main>
  );
}
"use client";
import { Dropzone,
DropzoneContent,
DropzoneEmptyState } from
"@/components/dropzone";
import { FieldInfo } from
"@/components/field-info";
import { Maps } from
"@/components/map";
import { NavBreadcrumb } from
"@/components/nav-breadcrumb";
import { Button } from
"@/components/ui/button";
import { Input } from
"@/components/ui/input";
import { Label } from
"@/components/ui/label";
import { Separator } from
"@/components/ui/separator";
import { Textarea } from
"@/components/ui/textarea";
import { useCourse } from
"@/hooks/use-course";
import { useSupabaseUpload }
from "@/hooks/use-supabase-
upload";
import { courseFormSchema,
TCourseForm } from
"@/schema/course-schemas";
import { useMapStore } from
"@/store/map-store";
import { useForm } from
"@tanstack/react-form";
import { ArrowLeftIcon,
Loader2 } from "lucide-react";
import { useRouter } from
"next/navigation";
import { toast } from
"sonner";

export default function Page()
{
  const { create } =
useCourse({ enableGet: false
});
  const { clickedMarker } =
useMapStore();
```

```
  const { back } =
useRouter();

  const imageUploadProps =
useSupabaseUpload({
    bucketName:
process.env.NEXT_PUBLIC_SUPABA
SE_BUCKET!,
    path: "test/images",
    allowedMimeTypes:
["image/*"],
    maxFiles: 1,
    maxFileSize: 1000 * 1000 *
20, // 20MB for the image
  });

  const videoUploadProps =
useSupabaseUpload({
    bucketName:
process.env.NEXT_PUBLIC_SUPABA
SE_BUCKET!,
    path: "test/videos",
    allowedMimeTypes:
["video/*"],
    maxFiles: 1,
    maxFileSize: 1000 * 1000 *
100,
  });

  // === Inisialisasi form ===
  const form = useForm({
    defaultValues: {
      name: "",
      price: "",
      address: "",
      phoneNumber: "",
      description: "",
      bankAccounts: [],
    } as TCourseForm,
    validators: { onSubmit:
courseFormSchema },
    onSubmit: async (values)
=> {
      console.log("Form data
before submission:", values);
// Debugging form data

      // Upload files to
Supabase
      const { success:
imageSuccess, error:
imageError } = await
imageUploadProps.onUpload();
      const { success:
videoSuccess, error:
```

```
videoError } = await
videoUploadProps.onUpload();

      // Show error if all
uploads fail
      if (imageSuccess.length
=== 0 || videoSuccess.length
=== 0) {
        toast.error("Upload
gagal. Silakan coba lagi.");
        return;
      }

      // Check if location is
selected
      if (clickedMarker ===
null) {
        toast.error("Silakan
pilih lokasi kursus");
        return;
      }

      // Show warning for
failed uploads
      if (imageError.length >
0 || videoError.length > 0) {

toast.warning(`Sebagian file
gagal diupload:
${[...imageError,
...videoError].map((e) =>
e.name).join(", ")}`);
      }

      // Prepare the paths of
successfully uploaded files
      const uploadedImagePaths
= imageSuccess.map((filename)
=> `test/images/${filename}`);
      const uploadedVideoPaths
= videoSuccess.map((filename)
=> `test/videos/${filename}`);

      // Send form data along
with uploaded paths to the
backend
      create.mutate({
        ...values.value,
        image:
uploadedImagePaths[0],
        video:
uploadedVideoPaths[0],
        longitude:
clickedMarker.longitude,
        latitude:
clickedMarker.latitude,
```

```
      });
    },
  });

  return (
    <form
      onSubmit={(e) => {
        e.preventDefault();
        e.stopPropagation();
        form.handleSubmit();
      }}
      className="space-y-8"
    >
      <NavBreadcrumb
current="/beranda/my-
course/create" />
      <section className="flex
items-center gap-3">
        <Button size="icon"
variant="outline"
className="rounded-full"
onClick={back}>
          <ArrowLeftIcon
size={18} />
        </Button>
        <h2 className="text-
2xl font-bold tracking-
tight">Buat Kursus</h2>
      </section>
      <Separator />
      <div className="h-
[400px]">
        <Maps
showScale={false} />
      </div>

      <div className="space-y-
3">
        <Label>Gambar</Label>
        <Dropzone
{...imageUploadProps}>
          <DropzoneEmptyState
/>
          <DropzoneContent />
        </Dropzone>
      </div>
      <div className="space-y-
3">
        <Label>Video </Label>
        <Dropzone
{...videoUploadProps}>
          <DropzoneEmptyState
/>
          <DropzoneContent />
        </Dropzone>
      </div>
```

```
      {/* Course Name */}
      <form.Field name="name">
        {(field) => (
          <div
className="space-y-3">

<Label>Name</Label>
            <Input

placeholder="Tulis judul
course..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Course Description
*/}
      <form.Field
name="description">
        {(field) => (
          <div
className="space-y-3">

<Label>Deskripsi</Label>
            <Textarea

placeholder="Tulis deskripsi
course..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
```

```
        )}
      </form.Field>

      {/* Course Address */}
      <form.Field
name="address">
        {(field) => (
          <div
className="space-y-3">

<Label>Alamat</Label>
            <Input

placeholder="Tulis alamat
kursus..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Course Phone Number
*/}
      <form.Field
name="phoneNumber">
        {(field) => (
          <div
className="space-y-3">
            <Label>Nomor
Telepon</Label>
            <Input

placeholder="Tulis nomor
telepon kursus..."
              id={field.name}

name={field.name}

value={field.state.value}

onBlur={field.handleBlur}
              onChange={(e) =>
field.handleChange(e.target.va
lue)}
            />
```

```
            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      {/* Account Owner */}
      <form.Field
name="bankAccounts">
        {(field) => (
          <div
className="space-y-3">
            <Label>Bank
Account</Label>

{field.state.value.map((bankAc
count, index) => (
              <div key={index}
className="flex gap-2 items-
center">
                <div
className="flex-1">
                  {/* Bank
Name */}
                  <Input

placeholder={`Nama Bank
${index + 1}`}
value={bankAccount.bankName}

onChange={(e) => {
                        const
updated =
[...field.state.value];

updated[index].bankName =
e.target.value;

field.handleChange(updated);
                      }}
                    />
                  </div>
                  <div
className="flex-1">
                    {/* Account
Number */}
                    <Input

placeholder={`Nomor Rekening
${index + 1}`}

value={bankAccount.accountNo}

onChange={(e) => {
```

```
                        const
updated =
[...field.state.value];

updated[index].accountNo =
e.target.value;

field.handleChange(updated);
                      }}
                    />
                  </div>
                  <div
className="flex-1">
                    {/* Owner
Name */}
                    <Input

placeholder={`Nama Pemilik
Rekening ${index + 1}`}

value={bankAccount.accountName
}

onChange={(e) => {
                        const
updated =
[...field.state.value];

updated[index].accountName =
e.target.value;

field.handleChange(updated);
                        }}
                      />
                    </div>

                    {/* Remove
Bank Account */}
                    <Button

type="button"

variant="outline"
                      size="icon"
                      onClick={()
=>
field.handleChange(field.state
.value.filter((_, i) => i !==
index))}
                    >
                      X
                    </Button>
                  </div>
                ))}
```

```
            {/* Add Bank
Account Button */}
            <Button
              type="button"
              size="sm"
variant="outline"
              onClick={() =>
field.handleChange([...field.s
tate.value, { bankName: "",
accountNo: "", accountName: ""
}])
              }
            >
              Tambah Bank
Account
            </Button>

            <FieldInfo
field={field} />
          </div>
        )}
      </form.Field>

      <Button
        className="w-full"
        type="submit"
disabled={create.isPending ||
imageUploadProps.loading ||
videoUploadProps.loading}
      >
        {(create.isPending ||
imageUploadProps.loading ||
videoUploadProps.loading) && (
          <Loader2
className="mr-2 h-4 w-4
animate-spin" />
        )}
        Submit
      </Button>
    </form>
  );
}
```

Lampiran-4 OrderActivity

```
"use client";
import { Empty } from
"@/components/empty";
import ListRenderer from
"@/components/list-renderer";
import { NavBreadcrumb } from
"@/components/nav-breadcrumb";
```

```
import { OrderItem,
OrderWithRelations } from
"@/components/order/item";
import { OrderItemCoursePaid }
from "@/components/order/item-
course-paid";
import { Input } from
"@/components/ui/input";
import { Tabs, TabsContent,
TabsList, TabsTrigger } from
"@/components/ui/tabs";
import { useGetOrderByUser }
from "@/hooks/use-order";
import { Search } from
"lucide-react";
import { useRouter,
useSearchParams } from
"next/navigation";
import { useDebouncedCallback
} from "use-debounce";

export default function Page()
{
  const params =
useSearchParams();
  const router = useRouter();

  const handleSearch =
useDebouncedCallback((value:
string) => {
    router.replace(`/list-
order?orderId=${value}`);
  }, 500);

  const orderId =
params.get("orderId") || "";

  const { data: orders,
isLoading: isOrdersLoading } =
useGetOrderByUser({ orderId
});
  const { data: ordersPaid,
isLoading: isOrdersPaidLoading
} = useGetOrderByUser({
status: "PAID" });

  return (
    <main className="space-y-
8">
      {/* Breadcrumb */}
      <NavBreadcrumb
current="/beranda/my-order" />

      {/* Header Section */}
```

```
      <section className="flex
flex-col md:flex-row gap-5
justify-between items-start
md:items-center">
        <h1 className="text-xl
sm:text-2xl font-bold">Orderan
Saya</h1>
      </section>

      <Tabs
defaultValue="orderan">
        <TabsList>
          <TabsTrigger
value="orderan">Orderan</TabsT
rigger>
          <TabsTrigger
value="kursus-terbeli">Kursus
Terbeli</TabsTrigger>
        </TabsList>
        <TabsContent
value="orderan">
          <div
className="space-y-6 mt-6">
            {/* Search Section
*/}
            <section
className="w-full max-w-md">
                <div
classNam="relative">
                  <Search
classNam="absolute left-3
top-1/2 transform -translate-
y-1/2 text-gray-400 h-4 w-4"
/>
                  <Input
                    type="text"

placeholder="Cari berdasarkan
Order ID..."

defaultValue={orderId}

onChange={(e) =>
handleSearch(e.target.value)}

classNam="pl-10"
                    />
                </div>
              </section>

<ListRenderer<OrderWithRelatio
ns>
                of={orders}
                render={(items)
=> <OrderItem item={items} />}
```

```
              className="grid
grid-cols-1 sm:grid-cols-2
lg:grid-cols-2 gap-8"

loadingState={<h1>Loading...</
h1>}

isLoading={isOrdersLoading}

emptyState={<Empty text="Tidak
ada orderan ditemukan" />}
            />
          </div>
        </TabsContent>
        <TabsContent
value="kursus-terbeli">
          <div
className="space-y-6 mt-6">

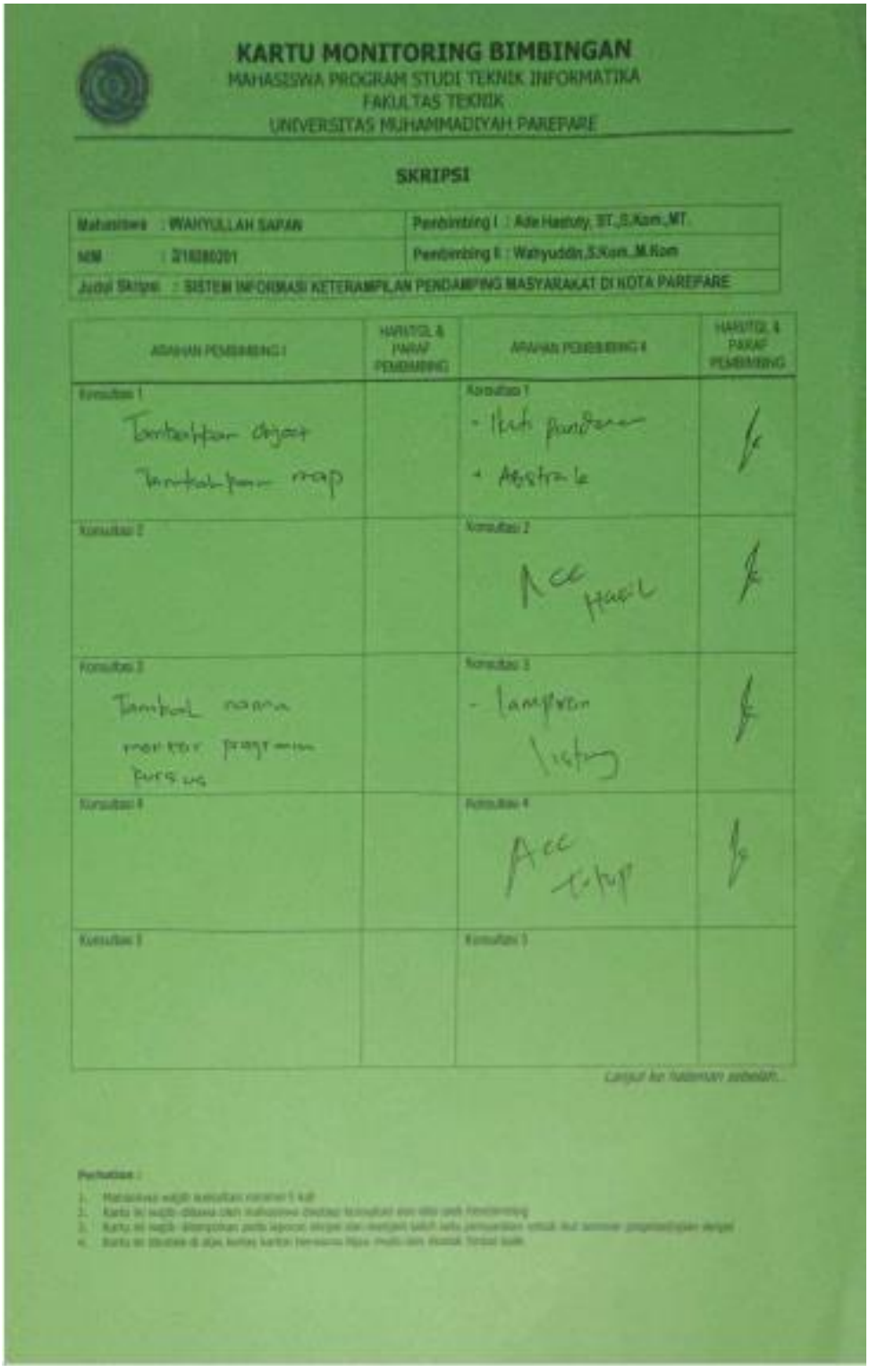
<ListRenderer<OrderWithRelatio
ns>
              of={ordersPaid}
              render={(items)
=> <OrderItemCoursePaid
item={items} />}
              className="grid
grid-cols-1 sm:grid-cols-2
lg:grid-cols-2 gap-4"

loadingState={<h1>Loading...</
h1>}

isLoading={isOrdersPaidLoading
}

emptyState={<Empty text="Tidak
ada kursus yang sudah dibeli"
/>}
            />
          </div>
        </TabsContent>
      </Tabs>
    </main>
  );
}
```

Lampiran-5 Motnitoring pembimbing



KARTU MONITORING BIMBINGAN
MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : WAHYULLAH SAFAN	Pembimbing I : Ade Hastuty, ST.,S.Kom.,MT.
NIM : 2108080201	Pembimbing II : Wahyuddin,S.Kom.,M.Kom
Judul Skripsi : SISTEM INFORMASI KETERAMPILAN PENDAMPING MASYARAKAT DI KOTA PAREPARE	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1 Tambahkan objek Tambahkan map		Konsultasi 1 - Hasil pembahasan - Abstrak	
Konsultasi 2		Konsultasi 2 Acc hasil	
Konsultasi 3 Tambah nama nomor program pursue		Konsultasi 3 - lampiran listing	
Konsultasi 4		Konsultasi 4 Acc tutup	
Konsultasi 5		Konsultasi 5	

Lanjut ke halaman selanjut...

Perhatikan :
1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan ada oleh pembimbing
3. Kartu ini wajib ditempelkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/skripsi/ujian
4. Kartu ini disimpan di atas kertas karton berwarna hijau muda dan ditolak 10 kali