Can you tell me what all changes we did?

Yes — here's a **clear, exact summary** of everything you changed for your part of the project.
You can paste this into a report or explain it in your team meeting.

✅ Changes You Made (Thread Structure & Deadline API)

You implemented the **deadline infrastructure** required for EDF scheduling.

1️⃣ Added `deadline` field to every thread

File: `thread.h`

Location: `struct thread`

✅ Added:

```c
int64_t deadline;    /* Absolute deadline in ticks */
```

✅ Meaning:

Now **every thread stores its own deadline**.
EDF uses this to decide which thread is most urgent.

2️⃣ Added a default deadline constant

File: `thread.h`

✅ Added:

```c
#define THREAD_NO_DEADLINE INT64_MAX
```

✅ Meaning:

Threads that do NOT specify a deadline behave like normal tasks.

EDF treats:

- smaller deadline = more urgent
- INT64_MAX = lowest priority (soft real-time or best-effort)

3️⃣ Initialized deadline during thread creation

File: `thread.c`

Location: `init_thread()`

✅ Added:

```c
t->deadline = THREAD_NO_DEADLINE;
```

✅ Meaning:

Every new thread starts with:

> "No real-time constraint unless explicitly assigned."

This prevents uninitialized data bugs.

4️⃣ Created deadline setter and getter API

File: `thread.h` and `thread.c`

✅ Implemented:

```c
void thread_set_deadline (int64_t deadline);
int64_t thread_get_deadline (void);
```

✅ Meaning:

Other kernel code can:

- change deadlines dynamically
- retrieve deadlines easily
- avoid touching struct internals directly

This makes EDF logic modular and clean.

5️⃣ Added deadline-aware thread creation

File: `thread.h` and `thread.c`

✅ Implemented:

```c
tid_t thread_create_with_deadline (...);
```

Internally:

- Calls `thread_create()`
- Finds the newly created thread by `tid`
- Assigns its deadline safely (interrupts off)

✅ Meaning:

Threads can now be created like:

```c
thread_create_with_deadline("jobA", PRI_DEFAULT, 500, fn, NULL);
```

So:

> Deadline is defined at birth.

✅ What You Did NOT Change

You:
✅ Did not modify the scheduler
✅ Did not change ready list logic
✅ Did not affect thread timing
✅ Did not break priority scheduling
✅ Did not alter thread switching logic

So:

> You added structure — not behavior.

✅ In One Sentence (for report or viva)

> “We extended PintOS threads with a deadline field, provided API support to set and get deadlines, initialized default behavior, and added a deadline-aware thread creation interface to enable EDF scheduling logic.”

✅ Your Contribution Status

You successfully implemented:

✅ Infrastructure layer
✅ Data model
✅ Kernel API
✅ Build verified
✅ No regressions

Your teammate now implements **EDF selection + preemption** on top.

If you want, I can also help you draft a **team report paragraph** or explain EDF flow in simple words.

Branched from OS project setup