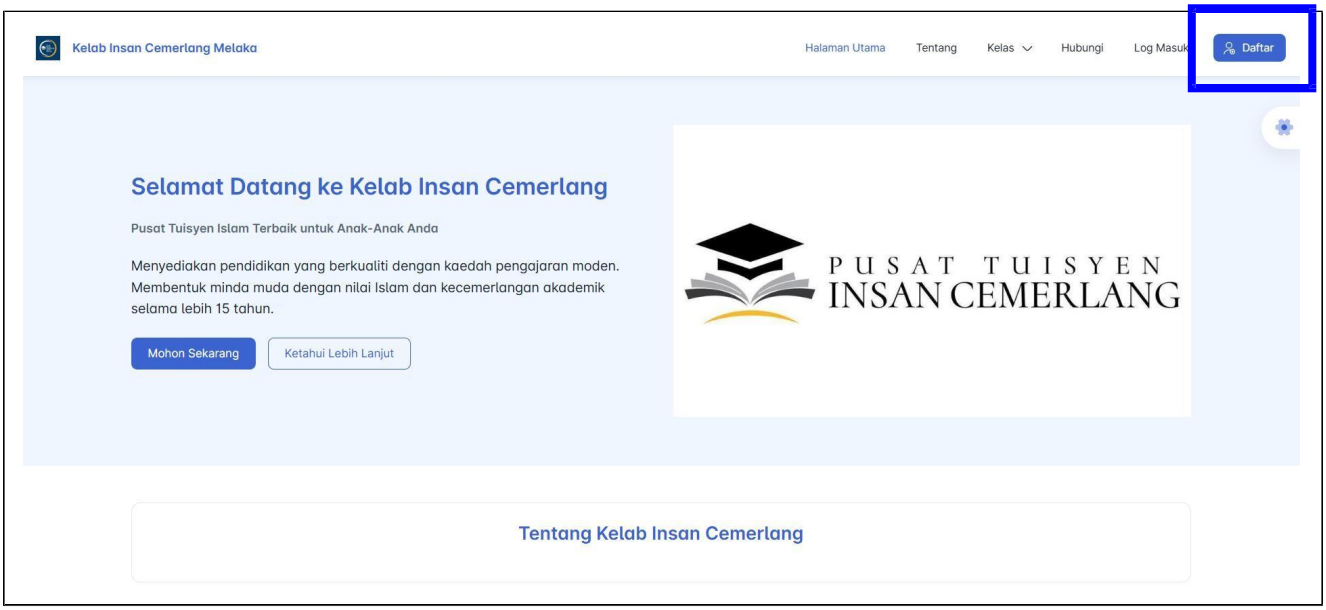
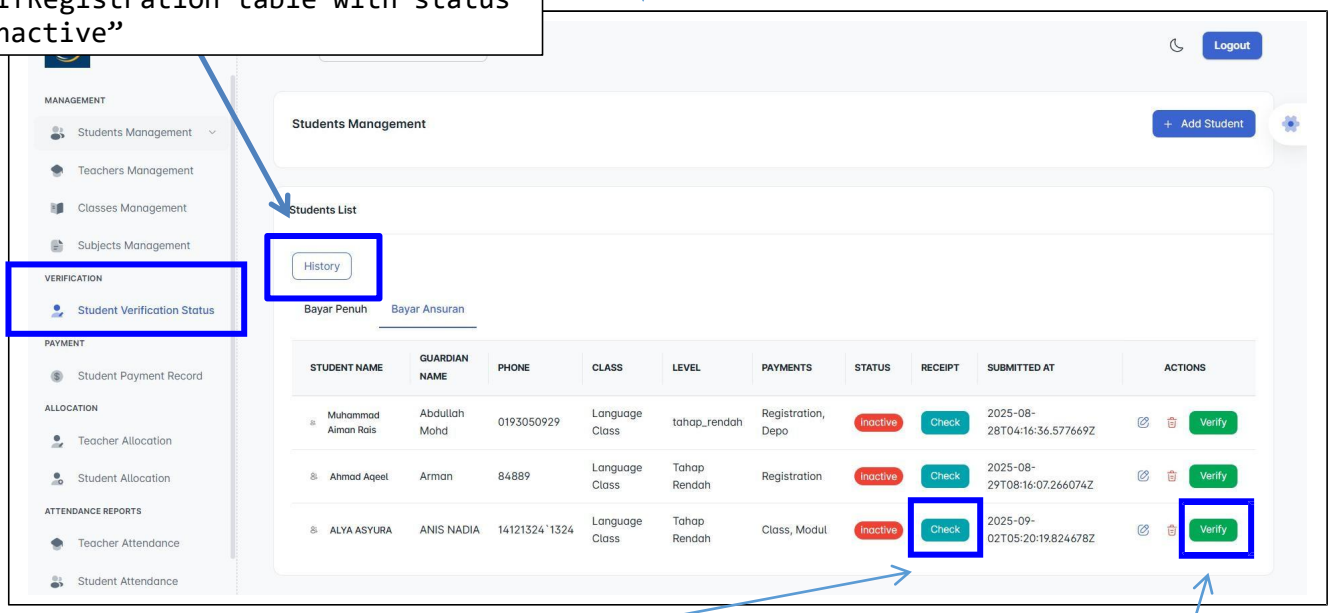
Client click here to do the SelfRegistration



Admin can view which student have verified before *which I took from SelfRegistration table with status “inactive”

After client registered . It will go to “Student Verification Status” - Admin Dashboard

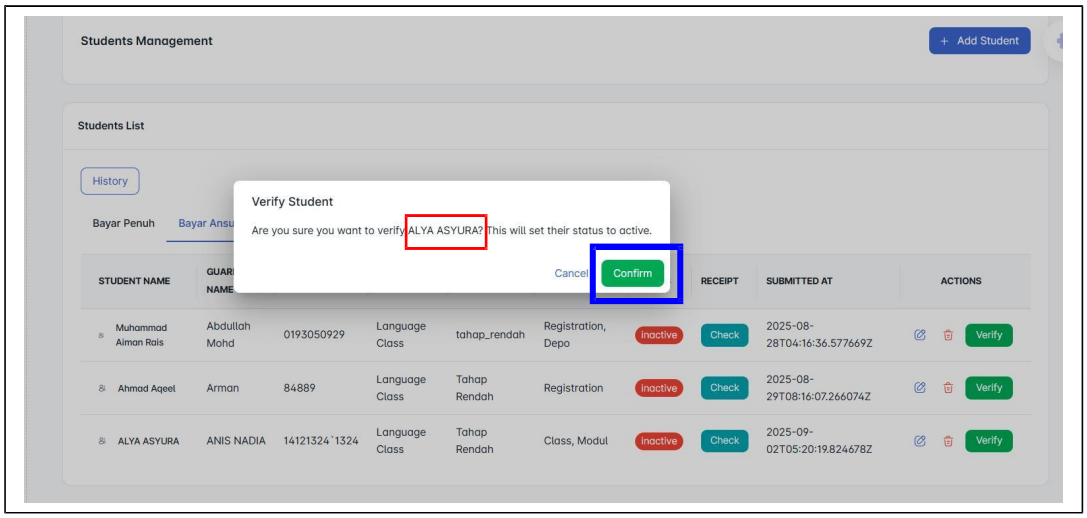


Admin can view reciept that uploaded from client . (Self-Registration)
They need to make sure the amount from the receipt and also make sure if there’s transactions from client or not.

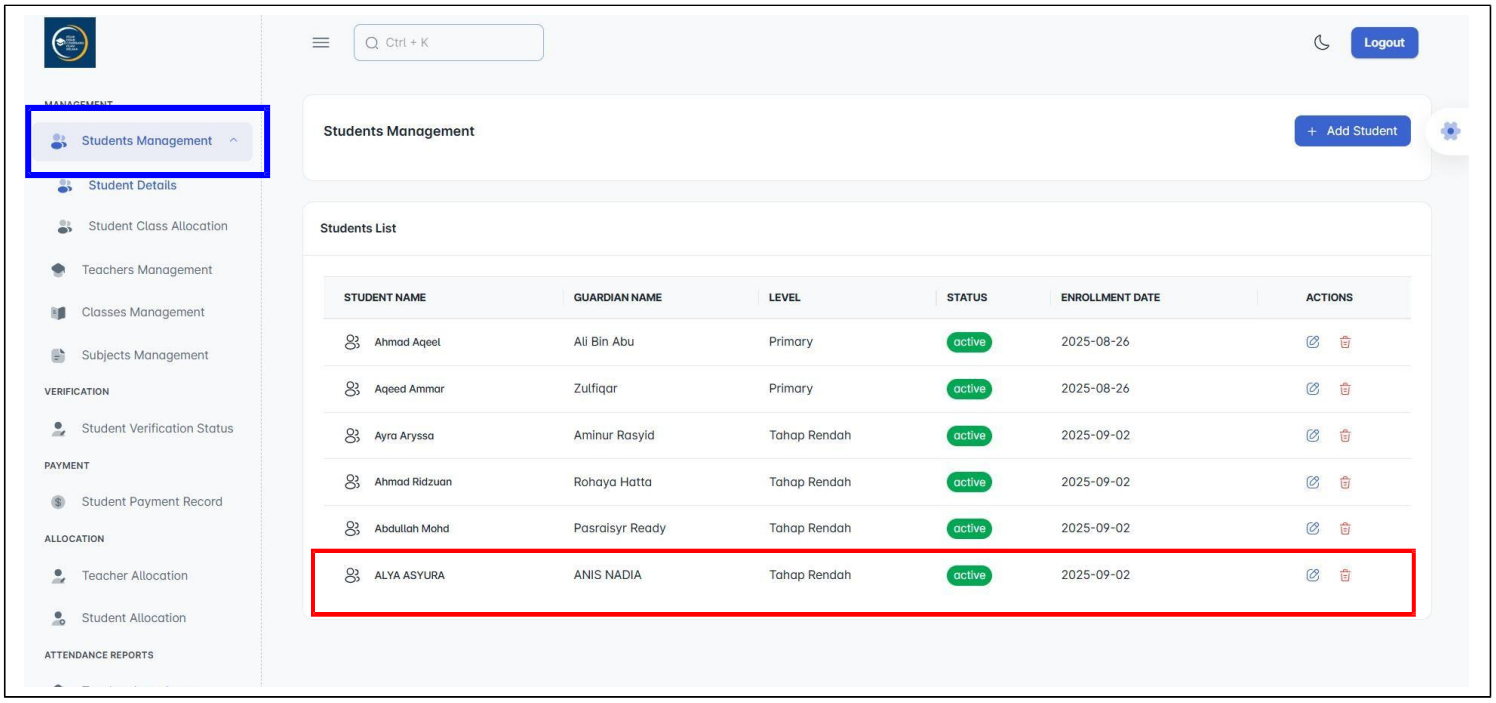
Admin can verify if they confirmed there’s transaction from client and received payment .

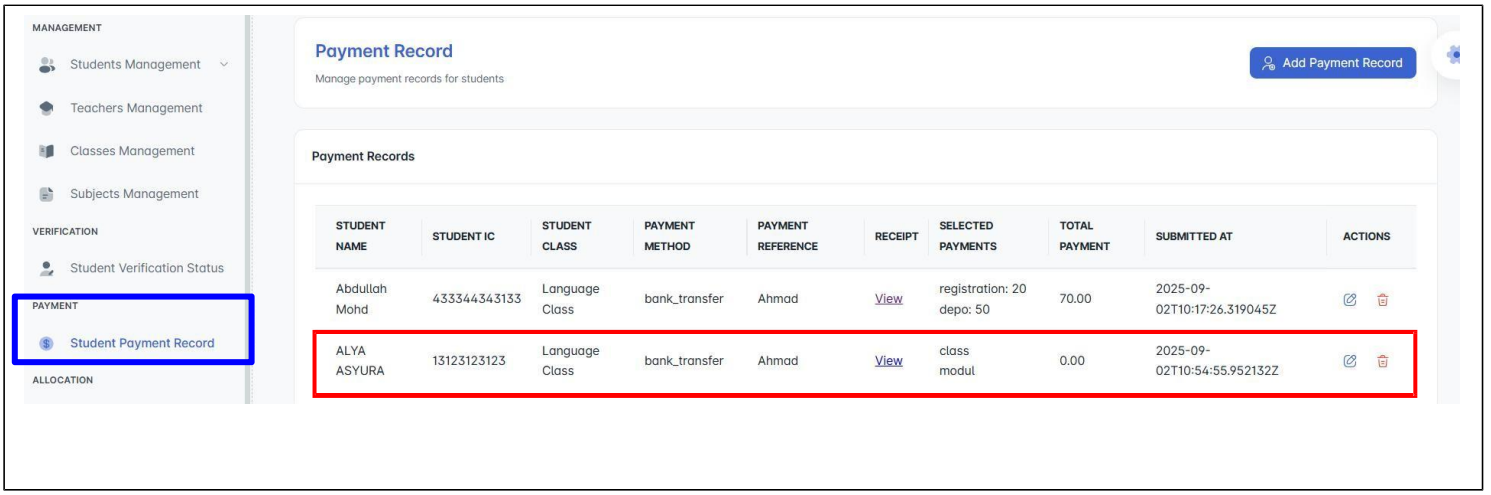
*Verify means change the status to “Active”.

it will automatically create and save in :
Student Table and Payment Table



After Admin confirm to verify :
It will automatically save
in : ”Student” and “Payment”





My backend models looks like : from SelfRegistration to Student and Payment

*I'm not an expert, but this is how the flow looks like.

*The admin can see which clients registered by themselves through the website.

```python
class SelfRegistration(models.Model):
    guardianName = models.CharField(max_length=100)
    guardianIC = models.CharField(max_length=20)
    guardianPhone = models.CharField(max_length=20)
    studentName = models.CharField(max_length=100)
    studentIC = models.CharField(max_length=20)
    address = models.TextField()
    studentClass  =  models.CharField(max_length=50)
    level  =  models.CharField(max_length=50)
    class_method = models.CharField(max_length=50)
    payment_method = models.CharField(max_length=50)
    payment_reference = models.CharField(max_length=100)
    payment_receipt = models.FileField(upload_to='receipts/')
    selected_payments = models.JSONField(default=list)  # Stores selected payment items
    total_payment = models.DecimalField(max_digits=10, decimal_places=2)  # Stores total
payment amount
    submitted_at = models.DateTimeField(auto_now_add=True)
    STATUS_CHOICES = [
        ('active', 'Active'),
        ('inactive', 'Inactive'),
    ]
    status = models.CharField(max_length=8, choices=STATUS_CHOICES, default='inactive')  #
Staff sets this

    def save(self, *args, **kwargs):
        was_inactive = False
        if self.pk:
            old = SelfRegistration.objects.get(pk=self.pk)
            was_inactive = old.status == 'inactive'
        super().save(*args, **kwargs)
        # If status changed to active, promote to Student
        if self.status == 'active' and (not self.pk or was_inactive):
            from ttp1_backend.models import Student, Class, Payment
            try:
                class_obj = Class.objects.get(id=self.studentClass)
            except Class.DoesNotExist:
                return
            Student.objects.get_or_create(
                studentIC=self.studentIC,
                defaults={
                    'guardianName': self.guardianName,
                    'guardianIC': self.guardianIC,
                    'guardianPhone': self.guardianPhone,
                    'studentName': self.studentName,
                    'address': self.address,
                    'level': self.level,
                    'status': 'active',
                    'enrollmentDate': self.submitted_at.date()
                }
            )
            Payment.objects.create( studentName=self.s
                tudentName, studentIC=self.studentIC,
                studentClass=class_obj.name if class_obj else None,
                payment_method=self.payment_method,
                payment_reference=self.payment_reference,
                payment_receipt=self.payment_receipt,
                selected_payments=self.selected_payments,
                total_payment=self.total_payment,
                submitted_at=self.submitted_at
            )
```

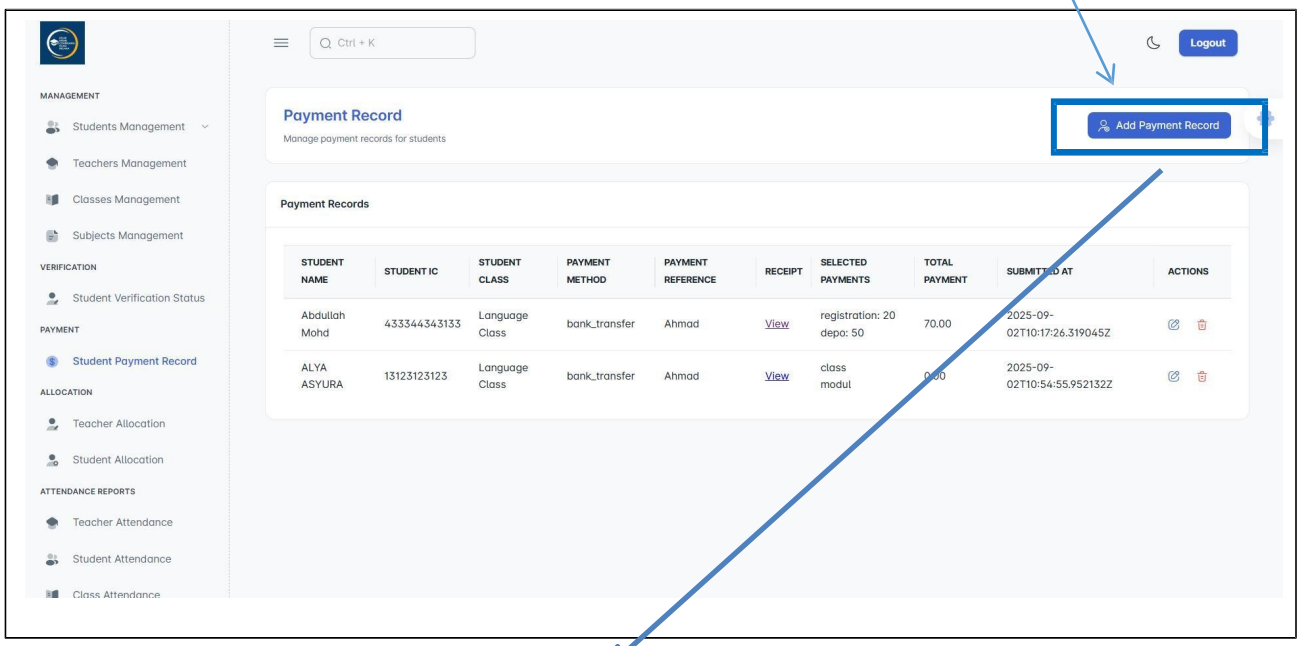
Class Payment

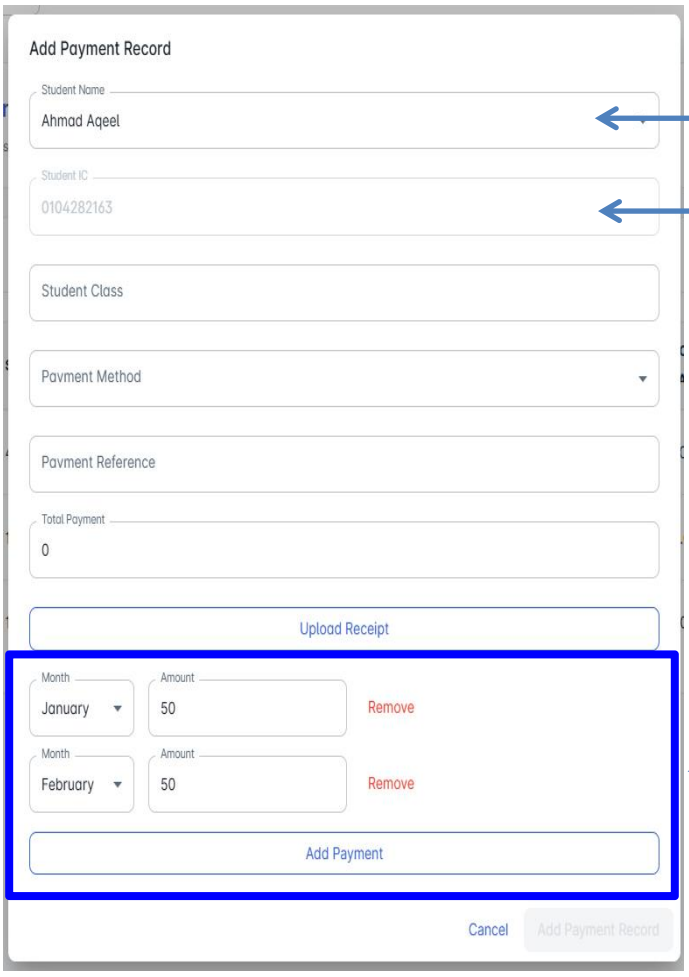
```
class Payment(models.Model):
    studentName = models.CharField(max_length=100)
    studentIC = models.CharField(max_length=20)
    studentClass = models.CharField(max_length=50, blank=True, null=True)
    payment_method = models.CharField(max_length=50)
    payment_reference = models.CharField(max_length=100)
    payment_receipt = models.FileField(upload_to='receipts/')
    selected_payments = models.JSONField(default=list)
    total_payment = models.DecimalField(max_digits=10, decimal_places=2,
default=0.00)
    submitted_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Payment {self.payment_reference} for {self.studentIC}"
```

Class Student

```
class Student(models.Model):
    STATUS_CHOICES = [
        ('active', 'Active'),
        ('inactive', 'Inactive'),
    ]
    guardianName = models.CharField(max_length=255)
    guardianIC = models.CharField(max_length=32)
    guardianPhone = models.CharField(max_length=32)
    studentName = models.CharField(max_length=255)
    studentIC = models.CharField(max_length=32)
    address = models.TextField()
    level = models.CharField(max_length=64)
    status = models.CharField(max_length=8, choices=STATUS_CHOICES,
default='active')
    enrollmentDate = models.DateField()

    def __str__(self):
        return self.studentName
```

Admin can add Payment by student for Monthly payment .
*Parents and Admin will contact by ws for montly payment. This is just for Admin to record and save in system.
*later on will make tabs that class them by “Payment Registartion” and “Monthly”
1. Payment Registration including : Depo , Modul , Registration Fee , Class
2. Monthly : (example : CLASS KAFA RM 50 , PAY RM 50 EVERYMONTH)



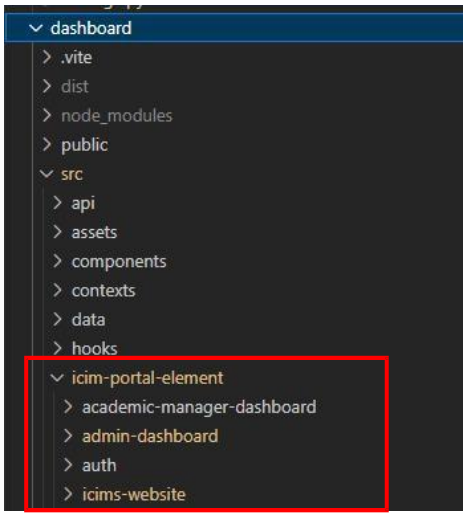


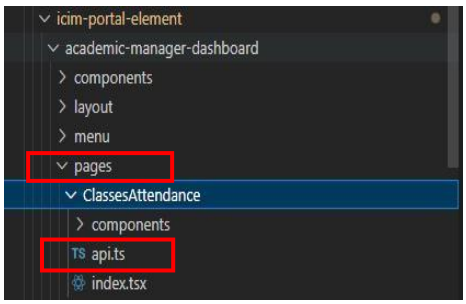
Fetch from Student

Auto-fill from selected student

JSON format that allows adding multiple months if a client wants to pay in advance for the upcoming months

Project ICIM directory :





*each pages have the api.ts itself

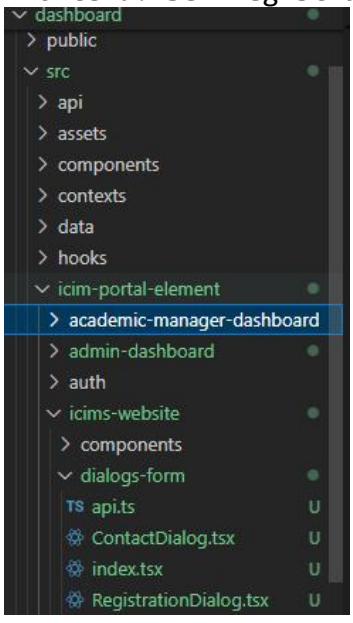
```
# Dashboard :
academic-manager-dashboard (same function with Admin)
admin-dashboard
teacher-dashboard

Frontend Folder :
icims-website
```

```
# every ["pages" folder] have their own parents and api.ts
```

```
example parents :
icim-portal-element > academic-manager-dashboard > pages > ClassesManagement > index.tsx
```

```
example api.ts:
icim-portal-element > academic-manager-dashboard > pages > ClassesManagement > api.ts
```

Frontend: SelfRegistration form



Icim-portal-element > icims-website> dialogs-form> api.ts