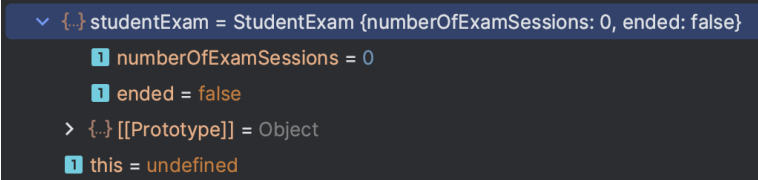
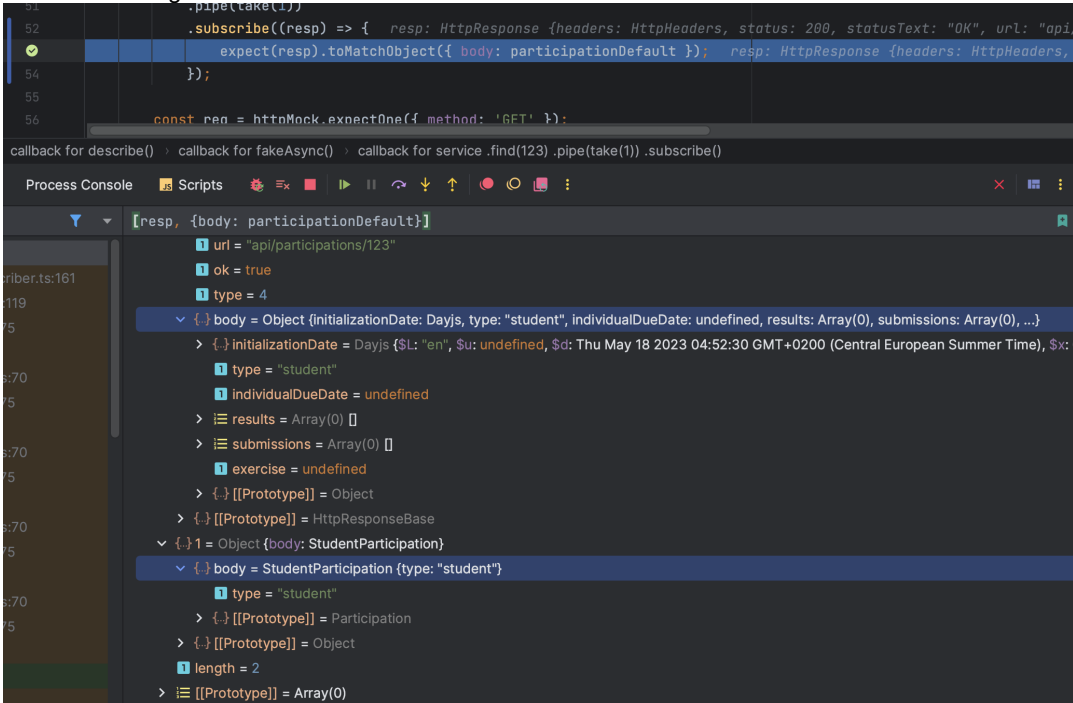
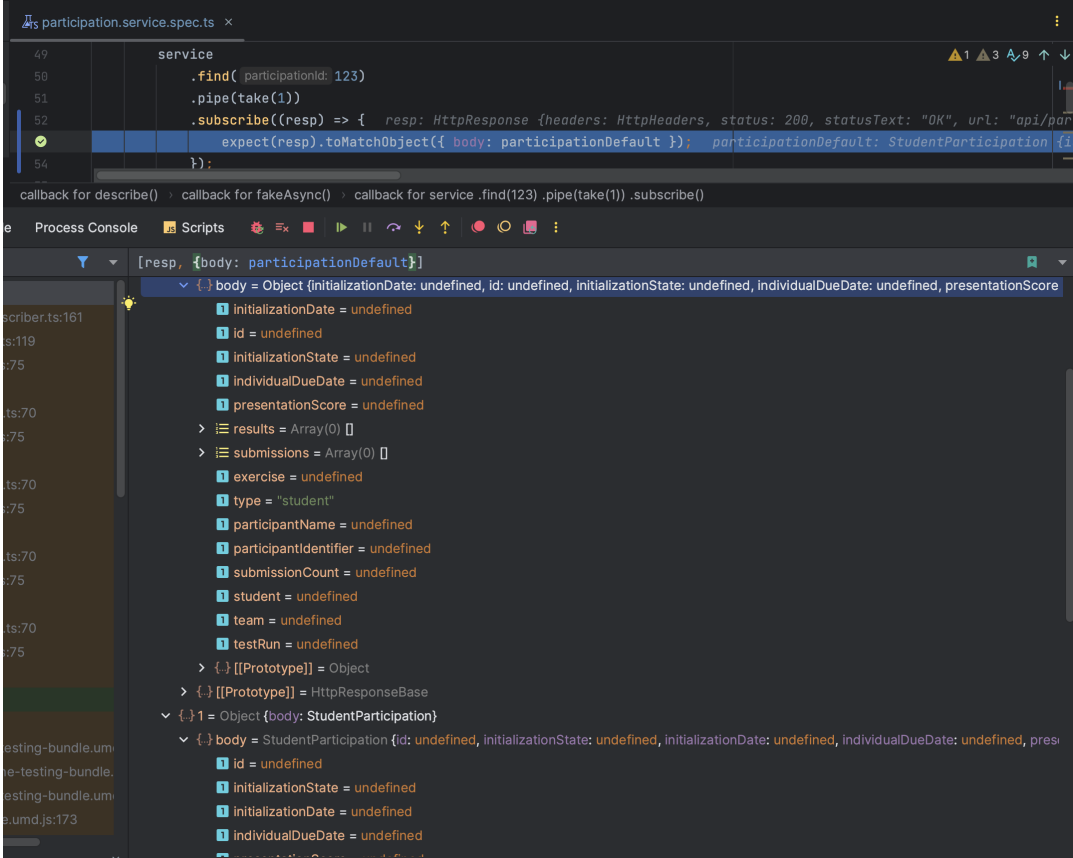


Component	Problem	Solution
assessment-locks.component.ts	P1: ngOnInit is an async function for no reason	Remove <code>async</code> from its signature
	P2: <code>courseId</code> is undefined in the component (because the subscription in <code>ngOnInit</code> isn't done when the method is called)	<code>ngOnInit</code> : only call <code>getAllLockedSubmissions</code> when there already is a subscription result
test-case-distribution-chart.component.ts	https://github.com/ls1intum/Artemis/pull/6557/files got lost in "fix" + merge	Apply the changes from the PR again
text-feedback-conflicts.component.ts	<code>this.activatedRoute.data</code> is undefined in <code>ngOnInit</code>	Proper mock for <code>activatedRoute</code>
	P2: mocked <code>activatedRoute</code> just doesn't have data, it's just mocked as <code>{ snapshot: { paramMap: convertToParamMap({ feedbackId: 1 }) } }</code> for some reason.	Proper mock for <code>activatedRoute</code>
navbar.component.ts	Angular changed the types for <code>RouterEvent</code> slightly ⇒ Type Error	Use <code>Event</code> instead of <code>RouterEvent</code> for the type, which seems to be the new name for this part
statistics-average-score-graph.component.ts / .html	P1: <code>typeFilter</code> is undefined when the HTML is rendered ⇒ crash	Merge <code>typeFilter</code> with <code>exerciseTypeFilter</code> , merge <code>categoryFilter</code> with <code>chartCategoryFilter</code>
	P2: <code>typeFilter</code> is set to <code>exerciseTypeFilter</code> outside of the constructor. I don't know what the order should be for this. Apparently, it changed. (<code>readonly typeFilter = this.exerciseTypeFilter;</code>)	Make the merged variables <code>readonly</code> directly
	P3: There is <code>typeFilter</code> and <code>exerciseTypeFilter</code> , although they are always the exact same reference (only one is private and the other is <code>readonly</code>)	
	P4: There is a typo in the <code>exerciseName</code> "FarcadePattern" (non-breaking)	Fix the typo
example-text-submission.component.spec.ts	<code>ngOnInit</code> is <code>async</code> . There is some problem with <code>fakeAsync</code> : Promises are not automatically awaited when calling <code>tick</code> (for some reason - <code>flush</code> / <code>flushMicrotasks</code> also does not work). It works when using <code>fakeAsync + async + await</code> .	Need control over <code>tick</code> time? ⇒ use <code>fakeAsync(async () => { ... })</code> and <code>await</code> Don't need it? ⇒ just use <code>async</code> tests Alternative solution (which I don't like as much): Don't use <code>async</code> Alternative solution (which I also don't like): Use

Component	Problem	Solution
		.then everywhere in your tests ⇒ replace <code>fakeAsync</code> with <code>async</code> directly
<code>text-submission-assessment.component.ts</code>	P1: <code>async-fakeAsync</code> problem, see <code>example-text-submission.component.spec.ts</code>	async tests
	P2: The mocked <code>ActivatedRoute</code> is passed as a function generating the route instead of a route??	Replace function with constant
	P3: Private <code>route</code> + <code>activeRoute</code> attributes of the component are set explicitly in the test instead of just using normal Angular DI	Properly mock instead of overriding private variables
	P4: <code>should navigate to conflicting submission test errors</code> because the <code>submission</code> is changed by another test before this test (doesn't error when started along). I guess that this was also the reason some of the other tests worked before, but shouldn't have worked.	Move initialization of used variables into <code>beforeEach</code>
	P5: <code>should display error when submitting but assessment invalid</code> fails because the component is not properly initialized when it is tested	<code>await component.ngInit()</code> in the <code>should display error when submitting but assessment invalid</code> test. I also added a test that the error is only shown once.
	P6: <code>should display error when complaint resolved but assessment invalid</code> fails because ????. I honestly don't understand what the test is supposed to actually do. I have the suspicion that the test only worked before because of some side effects from the other tests, but I don't know for sure.	I made the <code>assessment invalid</code> like it is done in the <code>should display error when saving but assessment invalid</code> test. This works & I'm ok with this after having sunk too much time into this issue. I've also added a check that the alert is only shown once, because I think that makes sense.
	P7: In the <code>should display error when complaint resolved but assessment invalid</code> , the <code>assessment</code> does not seem to be invalid at all	Make the <code>assessment invalid</code> by setting <code>component.unreferencedFeedback</code> manually
	P8: In the <code>should navigate to conflicting submission test</code> , the called URL is actually different. This is because <code>isExamMode</code> is <code>true</code> on the component. This makes sense, because the mocked route includes <code>examId</code> . Previously, the mocked route wasn't there, so it was pretty much empty.	Add the <code>exam</code> part to the expected URL because now, the mocked route assumes an exam exercise.
	Problem in general: The <code>textBlockRefs</code> , which is an essential part of the tested component(s) (because <code>referencedFeedback</code> is based on it), is exclusively modified in-place: <code>text-assessment-base.component.ts</code> , lines 118 and 136 (just for reference if you want to debug yourself). This makes it pretty hard to keep track of where data is coming from.	

Component	Problem	Solution
textblock-feedback-editor.component.ts	async-fakeAsync problem, see example-text-submission.component.ts	use async
tutorial-groups-registration-import-dialog.component.ts	P1: An error is logged to the console, although the promise result is caught	Add <code>jest.spyOn(console, 'error').mockImplementation();</code> so that the test is working. I added a comment explaining the situation. I'm relatively unhappy with this, but I cannot find a better working solution for this issue.
	P2: Angular Jest is having problems with async/await again. This time, a thrown error from a promise is recognized as a normal JS error instead of a promise rejection :(Add <code>jest.spyOn(console, 'error').mockImplementation();</code> so that the test is working. I added a comment explaining the situation. I'm relatively unhappy with this, but I cannot find a better working solution for this issue.
artemis-version.interceptor.ts	I've used 0.0.0 as the new default version number in <code>environment.ts</code> . The current test assumes 0.0.0 is not the version number from before and tests an update event with that. Therefore, there is no update.	I've changed the version in the test to <code>x.y.z</code> ⇒ it's different again
exam-participation.service.ts	The expected object is composed of <code>exercises</code> , <code>exam</code> and all properties of <code>examToSend</code> . However, the order of parameters in <code>Object.assign</code> is wrong. Additional info: Apparently, <code>new StudentExam()</code> created this before, which is wrong anyways. So the behavior is more correct than before: 	Change the order. Also, I removed an unnecessary copy of the data.
file-upload-exercise.service.ts	see exam-participation.service.ts	see exam-participation.service.ts
modeling-submission.service.ts	P1: Again, this is an issue with <code>new ___()</code> just having created an object with the set properties before: (see screenshot above)	Add <code>exerciseId = 5</code> to <code>elemDefault</code>

Component	Problem	Solution
	<p>P2: Also, the Jest <code>toMatchObject</code> check only checks equivalence for existing values on the object. Because the <code>exerciseId</code> wasn't existing on the object before, the check just passed</p>	<p>Used <code>toStrictEqual</code> instead of <code>toMatchObject</code></p>
	<p>P3: The <code>elemDefault</code> does not have an <code>exerciseId = 5</code> by default, but the comparison assumes so</p>	<p>Add <code>exerciseId = 5</code> to <code>elemDefault</code></p>
<p>participation.service.ts</p>	<p>The default value for new <code>Participation()</code> objects for <code>results</code> and <code>submission</code> is undefined, but the <code>participation.service</code> has functions mapping empty arrays to them, so the expectations don't match here. Apparently, this issue came up only now because the way how empty new JS objects are initialized now: <code>new StudentParticipation()</code> now creates an object that's initialized with a lot of undefined values, which breaks the test using <code>.toMatchObject</code>. Here's a debugger breakpoint from before the changes:</p>  <pre> 52 .pipe(take(1)) 53 .subscribe((resp) => { resp: HttpResponse {headers: HttpHeaders, status: 200, statusText: "OK", url: "api/participations/123"} 54 expect(resp).toMatchObject({ body: participationDefault }); 55 }); 56 const req = httpMock.expectOne({ method: 'GET' }); </pre> <p>Process Console Scripts [resp, {body: participationDefault}]</p> <ul style="list-style-type: none"> url = "api/participations/123" ok = true type = 4 body = Object {initializationDate: Days, type: "student", individualDueDate: undefined, results: Array(0), submissions: Array(0), ...} <ul style="list-style-type: none"> initializationDate = Days {\$_: "en", \$u: undefined, \$d: Thu May 18 2023 04:52:30 GMT+0200 (Central European Summer Time), \$x: ...} type = "student" individualDueDate = undefined results = Array(0) [] submissions = Array(0) [] exercise = undefined [[Prototype]] = Object [[Prototype]] = HttpResponseBase 1 = Object {body: StudentParticipation} <ul style="list-style-type: none"> body = StudentParticipation {type: "student"} <ul style="list-style-type: none"> type = "student" [[Prototype]] = Participation [[Prototype]] = Object length = 2 [[Prototype]] = Array(0) 	<p>initialize <code>StudentParticipation</code> so that it's compatible with <code>.toMatchObject</code> expectations</p>

Component	Problem	Solution
	<p>- After, without any changes to the test code:</p>  <pre> participation.service.spec.ts 49 service 50 .find(participationId: 123) 51 .pipe(take(1)) 52 .subscribe((resp) => { resp: HttpResponse {headers: HttpHeaders, status: 200, statusText: "OK", url: "api/participation/123"} 53 expect(resp).toMatchObject({ body: participationDefault }); participationDefault: StudentParticipation {id: 123, initializationDate: undefined, initializationState: undefined, individualDueDate: undefined, presentationScore: undefined, results: Array(0) [], submissions: Array(0) [], exercise: undefined, type: "student", participantName: undefined, participantIdentifier: undefined, submissionCount: undefined, student: undefined, team: undefined, testRun: undefined} 54 }); </pre> <p>callstack: callback for describe() > callback for fakeAsync() > callback for service.find(123).pipe(take(1)).subscribe()</p> <pre> [resp, {body: participationDefault}] body = Object {initializationDate: undefined, id: undefined, initializationState: undefined, individualDueDate: undefined, presentationScore: undefined, results: Array(0) [], submissions: Array(0) [], exercise: undefined, type: "student", participantName: undefined, participantIdentifier: undefined, submissionCount: undefined, student: undefined, team: undefined, testRun: undefined} [[Prototype]] = Object [[Prototype]] = HttpResponseBase 1 = Object {body: StudentParticipation {id: undefined, initializationState: undefined, initializationDate: undefined, individualDueDate: undefined, presentationScore: undefined, results: Array(0) [], submissions: Array(0) [], exercise: undefined, type: "student", participantName: undefined, participantIdentifier: undefined, submissionCount: undefined, student: undefined, team: undefined, testRun: undefined}} body = StudentParticipation {id: undefined, initializationState: undefined, initializationDate: undefined, individualDueDate: undefined, presentationScore: undefined, results: Array(0) [], submissions: Array(0) [], exercise: undefined, type: "student", participantName: undefined, participantIdentifier: undefined, submissionCount: undefined, student: undefined, team: undefined, testRun: undefined} id = undefined initializationState = undefined initializationDate = undefined individualDueDate = undefined </pre>	
programming-exercise.service.ts	see exam-participation.service.ts	see exam-participation.service.ts
programming-exercise.service.ts	see exam-participation.service.ts	see exam-participation.service.ts
navigation-util.service.ts	P1: The value of urlTreeMock isn't an input anywhere, it's just an expectation. I don't know how the router would "know" this value.	Add the mockReturnValue

Component	Problem	Solution
	P2: Apparently, the <code>MockRouter</code> uses "testValue" as its default return value, which probably is why the tests passed before. This makes the test more likely to pass "accidentally".	Change the test value to "uriTreeMockTestValue", to ensure that the values are not accidentally mixed up.
	P3: The tests around the interaction between <code>router.createUrlTree</code> and <code>router.serializeUrl</code> is kind of useless when the mock router always returns a constant: <code>serializeUrl = jest.fn().mockReturnValue('testValue');</code>	Mock the return value of <code>serializationMock</code> as well and check that (params are already checked)