

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

NGÀNH CÔNG NGHỆ THÔNG TIN



BÁO CÁO KIỂM THỦ PHẦN MỀM

**ĐỀ TÀI: KIỂM THỦ WEBSITE BÁN SÁCH VÀ
THIẾT BỊ HỌC TẬP**

GVHD: ThS. Nguyễn Trần Thị Văn

SVTH: Bùi Hà Nhi 18110168

Hoàng Minh Quang 18110181

Nguyễn Quang Vũ 18110241

TP.HCM, tháng 12 năm 2021

LỜI CẢM ƠN

Để hoàn thành tốt đề tài và bài báo cáo này, chúng em xin gửi lời cảm ơn chân thành đến giảng viên, thầy Nguyễn Trần Thị Văn, người đã trực tiếp hỗ trợ chúng em trong suốt quá trình làm đề tài. Chúng em cảm ơn thầy đã đưa ra những lời khuyên từ kinh nghiệm thực tiễn của mình để định hướng cho chúng em đi đúng với yêu cầu của đề tài đã chọn, luôn giải đáp thắc mắc và đưa ra những góp ý, chỉnh sửa kịp thời giúp chúng em khắc phục nhược điểm và hoàn thành tốt cũng như đúng thời hạn đã đề ra.

Chúng em cũng xin gửi lời cảm ơn chân thành các quý thầy cô trong khoa Đào tạo Chất Lượng Cao nói chung và ngành Công Nghệ Thông Tin nói riêng đã tận tình truyền đạt những kiến thức cần thiết giúp chúng em có nền tảng để làm nên đề tài này, đã tạo điều kiện để chúng em có thể tìm hiểu và thực hiện tốt đề tài. Cùng với đó, chúng em xin được gửi cảm ơn đến các bạn cùng khóa đã cung cấp nhiều thông tin và kiến thức hữu ích giúp chúng em có thể hoàn thiện hơn đề tài của mình.

Đề tài và bài báo cáo được chúng em thực hiện trong khoảng thời gian ngắn, với những kiến thức còn hạn chế cùng nhiều hạn chế khác về mặt kỹ thuật và kinh nghiệm trong việc thực hiện một dự án phần mềm. Do đó, trong quá trình làm nên đề tài có những thiếu sót là điều không thể tránh khỏi nên chúng em rất mong nhận được những ý kiến đóng góp quý báu của các quý thầy cô để kiến thức của chúng em được hoàn thiện hơn và chúng em có thể làm tốt hơn nữa trong những lần sau.

Chúng em xin chân thành cảm ơn. Cuối lời, chúng em kính chúc quý thầy, quý cô luôn dồi dào sức khỏe và thành công hơn nữa trong sự nghiệp trồng người. Một lần nữa chúng em xin chân thành cảm ơn.

Tp. Hồ Chí Minh, tháng 12 năm 2021

MỤC LỤC

LỜI CẢM ƠN.....	12
PHIẾU NHẬN XÉT CỦA GIÁO VIÊN.....	35
BẢNG PHÂN CÔNG CÔNG VIỆC.....	37
CHƯƠNG I: TEST PLAN.....	39
1. Tổng quan.....	39
1.1. Thông tin dự án.....	39
1.2. Tổng quan dự án.....	39
1.3. Thực trạng.....	40
1.4. Định nghĩa vấn đề.....	40
1.5. Giải pháp đề xuất.....	40
1.5.1 <i>Chức năng</i>	41
1.5.2 <i>Ưu điểm và nhược điểm</i>	41
1.6. Các yêu cầu về chức năng.....	42
2. Test plan.....	42
2.1. Mục tiêu kiểm thử.....	42
2.2. Nguyên tắc kiểm tra.....	43
2.3. Phương pháp tiếp cận dữ liệu.....	43
2.4. Phạm vi và mức độ kiểm tra.....	43
2.4.1. <i>Khám phá</i>	44
2.4.2. <i>Kiểm thử chức năng</i>	44
2.4.3. <i>Kiểm tra sự chấp nhận của người dùng (UAT)</i>	45
3. Chiến lược thực hiện.....	45
3.1. Tiêu chí đầu vào và đầu ra.....	45
3.2. Chu kỳ kiểm tra.....	46
3.3. Xác thực và quản lý sai sót.....	47
3.4. Số liệu kiểm tra.....	47
CHƯƠNG II: CÁC PHƯƠNG PHÁP KIỂM THỬ.....	49
1. Kiểm thử đơn vị – Unit test.....	49
2. Kiểm thử tích hợp – Intergration Test.....	50
3. Kiểm thử hệ thống – System Test.....	51
4. Kiểm thử chấp nhận sản phẩm – Acceptance Test.....	52
● Alpha & Beta Testing.....	52
● Contract Acceptance Testing.....	52
● Regulation Acceptance Testing.....	52
● Operational acceptance Testing.....	52
● Black Box Testing.....	52

CHƯƠNG III: MÔ HÌNH HOÁ YÊU CẦU.....	53
1. Usecase diagram.....	53
2. Chi tiết use case.....	54
2.1. Register.....	54
2.2. Login.....	57
2.3. Forgot Password.....	59
2.4. Reset Password.....	61
2.5. See All Products.....	63
2.6. See A Product Details.....	65
2.7. Contact to Group through email.....	67
2.8. Search Products by Keyword.....	69
2.9. Filter Products.....	71
2.10. Log out.....	73
2.11. See Profile Details.....	76
2.12. Update Profile.....	77
2.13. Update Password.....	79
2.14. Create A Product Review.....	82
2.15. Add to Cart.....	84
2.16. Take an order.....	86
2.17. Check out.....	88
2.18. See all orders.....	91
2.19. See a single order.....	93
2.20. See All Users.....	95
2.21. Update User Role.....	97
2.22. Delete Users.....	99
2.23. See All Products.....	101
2.24. Create Products.....	103
2.25. Update Products.....	107
2.26. Delete Products.....	109
2.27. See All Orders on system.....	111
2.28. Update An Order.....	113
2.29. Delete An Order.....	115
2.30. See All Reviews of a Product.....	117
2.31. Delete A Product Review.....	119
CHƯƠNG IV: GIAO DIỆN -.....	122
HƯỚNG DẪN SỬ DỤNG.....	122
1. Giao diện trang chủ.....	122

2. Trang đăng nhập, đăng ký.....	122
3. Giao diện trang sản phẩm theo danh mục sản phẩm.....	123
4. Giao diện trang chi tiết sản phẩm.....	124
5. Giỏ hàng.....	125
6. Thông tin mua hàng, xác nhận thanh toán.....	125
7. Tìm kiếm.....	128
8. Giao diện lọc theo giá hoặc review sản phẩm.....	129
9. Liên hệ.....	129
10. Giao diện và chức năng admin.....	130
CHƯƠNG V: ĐỒ THỊ DÒNG ĐIỀU KHIỂN.....	137
1. User - Register.....	137
2. User - Login.....	137
3. User - Forgot Password.....	138
4. User - Reset Password.....	140
5. User - See All Products.....	141
6. User - See A Product Details.....	142
10. Customer - Log out.....	143
11. Customer - See Profile Details.....	143
12. Customer - Update Profile.....	144
13. Customer - Update Password.....	145
14. Customer - Create A Product Review.....	146
16. Customer - Take an order.....	148
17. Customer - Check out.....	149
18. Customer - See all orders.....	149
19. Customer - See a single order.....	150
20. Admin - See All Users.....	151
21. Admin - Update User Role.....	152
22. Admin - Delete Users.....	152
23. Admin - See All Products.....	153
24. Admin - Create Products.....	154
25. Admin - Update Products.....	155
26. Admin - Delete Products.....	157
27. Admin - See All Orders on system.....	158
28. Admin - Update An Order.....	159
29. Admin - Delete An Order.....	160
30. Admin - See All Reviews of a Product.....	161
31. Admin - Delete A Product Review.....	161

CHƯƠNG VI: ĐỒ THỊ DÒNG DỮ LIỆU - KIỂM THỬ ĐỜI SỐNG CÁC BIẾN.164

1. User - Register.....	164
- Kiểm thử đời sống biến a(req):.....	165
- Kiểm thử đời sống biến b(res):.....	165
- Kiểm thử đời sống biến c(next):.....	166
- Kiểm thử đời sống biến d(myCloud):.....	166
- Kiểm thử đời sống biến e(avatar):.....	167
- Kiểm thử đời sống biến f(name):.....	167
- Kiểm thử đời sống biến g(email):.....	168
- Kiểm thử đời sống biến h(password):.....	168
- Kiểm thử đời sống biến i(user):.....	169
2. User - Login.....	169
- Kiểm thử đời sống biến a(req):.....	170
- Kiểm thử đời sống biến b(res):.....	171
- Kiểm thử đời sống biến c(next):.....	171
- Kiểm thử đời sống biến d(email):.....	172
- Kiểm thử đời sống biến e(password):.....	173
- Kiểm thử đời sống biến f(user):.....	173
- Kiểm thử đời sống biến g(isPasswordMatched):.....	174
3. User - Forgot Password.....	174
- Kiểm thử đời sống biến a(req):.....	176
- Kiểm thử đời sống biến b(res):.....	177
- Kiểm thử đời sống biến c(next):.....	178
- Kiểm thử đời sống biến d(user):.....	179
- Kiểm thử đời sống biến e(email):.....	180
- Kiểm thử đời sống biến f(resetToken):.....	181
- Kiểm thử đời sống biến g(resetPasswordUrl):.....	182
- Kiểm thử đời sống biến h(message):.....	183
- Kiểm thử đời sống biến i(subject):.....	184
- Kiểm thử đời sống biến k(error):.....	185
- Kiểm thử đời sống biến l(resetPasswordToken):.....	186
- Kiểm thử đời sống biến m(resetPasswordExpire):.....	187
4. User - Reset Password.....	187
- Kiểm thử đời sống biến a(req):.....	189
- Kiểm thử đời sống biến b(res):.....	189
- Kiểm thử đời sống biến c(next):.....	190
- Kiểm thử đời sống biến d(resetPasswordToken):.....	191

- Kiểm thử đời sống biến e(token):.....	191
- Kiểm thử đời sống biến f(user):.....	192
- Kiểm thử đời sống biến g(resetPasswordExpire):.....	193
- Kiểm thử đời sống biến h(confirmPassword):.....	193
- Kiểm thử đời sống biến i(password):.....	194
5. User - See All Products.....	194
- Kiểm thử đời sống biến a(req):.....	196
- Kiểm thử đời sống biến b(res):.....	196
- Kiểm thử đời sống biến c(next):.....	197
- Kiểm thử đời sống biến d(resultPerPage):.....	197
- Kiểm thử đời sống biến e(productsCount):.....	198
- Kiểm thử đời sống biến f(Product):.....	198
- Kiểm thử đời sống biến g(apiFeature):.....	199
- Kiểm thử đời sống biến h(products):.....	199
- Kiểm thử đời sống biến i(filteredProductsCount):.....	200
- Kiểm thử đời sống biến hk(success):.....	201
6. User - See A Product Details.....	201
- Kiểm thử đời sống biến a(req):.....	202
- Kiểm thử đời sống biến b(res):.....	202
- Kiểm thử đời sống biến c(next):.....	202
- Kiểm thử đời sống biến d(product):.....	203
- Kiểm thử đời sống biến e(Product):.....	203
- Kiểm thử đời sống biến f(id):.....	203
- Kiểm thử đời sống biến g(success):.....	204
10. Customer - Log out.....	204
- Kiểm thử đời sống biến a(req):.....	205
- Kiểm thử đời sống biến b(res):.....	205
- Kiểm thử đời sống biến c(next):.....	205
- Kiểm thử đời sống biến d(expires):.....	206
- Kiểm thử đời sống biến e(httpOnly):.....	206
- Kiểm thử đời sống biến f(success):.....	206
- Kiểm thử đời sống biến g(message):.....	207
11. Customer - See Profile Details.....	207
- Kiểm thử đời sống biến a(req):.....	207
- Kiểm thử đời sống biến b(res):.....	208
- Kiểm thử đời sống biến c(next):.....	208
- Kiểm thử đời sống biến d(user):.....	208

- Kiểm thử đòi hỏi biến e(User):.....	209
- Kiểm thử đòi hỏi biến f(id):.....	209
- Kiểm thử đòi hỏi biến g(success):.....	209
12. Customer - Update Profile.....	209
- Kiểm thử đòi hỏi biến a(req):.....	211
- Kiểm thử đòi hỏi biến b(res):.....	211
- Kiểm thử đòi hỏi biến c(next):.....	212
- Kiểm thử đòi hỏi biến d(newUserData):.....	212
- Kiểm thử đòi hỏi biến e(name):.....	213
- Kiểm thử đòi hỏi biến f(email):.....	213
- Kiểm thử đòi hỏi biến g(avatar):.....	214
- Kiểm thử đòi hỏi biến h(user):.....	214
- Kiểm thử đòi hỏi biến i(User):.....	215
- Kiểm thử đòi hỏi biến k(imageId):.....	215
- Kiểm thử đòi hỏi biến l(myCloud):.....	216
- Kiểm thử đòi hỏi biến m(success):.....	216
13. Customer - Update Password.....	216
- Kiểm thử đòi hỏi biến a(req):.....	218
- Kiểm thử đòi hỏi biến b(res):.....	218
- Kiểm thử đòi hỏi biến c(next):.....	219
- Kiểm thử đòi hỏi biến d(user):.....	220
- Kiểm thử đòi hỏi biến e(User):.....	220
- Kiểm thử đòi hỏi biến f(id):.....	221
- Kiểm thử đòi hỏi biến g(password):.....	222
- Kiểm thử đòi hỏi biến h(isPasswordMatched):.....	222
- Kiểm thử đòi hỏi biến i(oldPassword):.....	223
- Kiểm thử đòi hỏi biến k(newPassword):.....	224
- Kiểm thử đòi hỏi biến l(confirmPassword):.....	224
14. Customer - Create A Product Review.....	225
- Kiểm thử đòi hỏi biến a(req):.....	227
- Kiểm thử đòi hỏi biến b(res):.....	228
- Kiểm thử đòi hỏi biến c(next):.....	229
- Kiểm thử đòi hỏi biến d(rating):.....	230
- Kiểm thử đòi hỏi biến e(comment):.....	231
- Kiểm thử đòi hỏi biến f(productId):.....	232
- Kiểm thử đòi hỏi biến g(review):.....	233
- Kiểm thử đòi hỏi biến h(user):.....	234

- Kiểm thử đời sóng biến i(name):.....	235
- Kiểm thử đời sóng biến k(product):.....	236
- Kiểm thử đời sóng biến l(Product):.....	237
- Kiểm thử đời sóng biến m(isReviewed):.....	238
- Kiểm thử đời sóng biến n(reviews):.....	239
- Kiểm thử đời sóng biến o(rev):.....	240
- Kiểm thử đời sóng biến p(avg):.....	241
- Kiểm thử đời sóng biến r(ratings):.....	242
- Kiểm thử đời sóng biến s(success):.....	243
16. Customer - Take an order.....	243
- Kiểm thử đời sóng biến a(req):.....	245
- Kiểm thử đời sóng biến b(res):.....	245
- Kiểm thử đời sóng biến c(next):.....	245
- Kiểm thử đời sóng biến d(shippingInfo):.....	246
- Kiểm thử đời sóng biến e(orderItems):.....	246
- Kiểm thử đời sóng biến f(paymentInfo):.....	246
- Kiểm thử đời sóng biến g(itemsPrice):.....	247
- Kiểm thử đời sóng biến h(taxPrice):.....	247
- Kiểm thử đời sóng biến i(shippingPrice):.....	247
- Kiểm thử đời sóng biến k(totalPrice):.....	248
- Kiểm thử đời sóng biến l(order):.....	248
- Kiểm thử đời sóng biến m(Order):.....	248
- Kiểm thử đời sóng biến n(paidAt):.....	249
- Kiểm thử đời sóng biến o(user):.....	249
- Kiểm thử đời sóng biến p(success):.....	249
17. Customer - Check out.....	250
- Kiểm thử đời sóng biến a(req):.....	250
- Kiểm thử đời sóng biến b(res):.....	251
- Kiểm thử đời sóng biến c(next):.....	251
- Kiểm thử đời sóng biến d(myPayment):.....	251
- Kiểm thử đời sóng biến e(stripe):.....	252
- Kiểm thử đời sóng biến f(paymentIntents):.....	252
- Kiểm thử đời sóng biến g(amount):.....	252
- Kiểm thử đời sóng biến h(currency):.....	253
- Kiểm thử đời sóng biến i(metadata):.....	253
- Kiểm thử đời sóng biến k(company):.....	253
- Kiểm thử đời sóng biến l(success):.....	254

18. Customer - See all orders.....	254
- Kiểm thử đời sống biến a(req):.....	255
- Kiểm thử đời sống biến b(res):.....	255
- Kiểm thử đời sống biến c(next):.....	255
- Kiểm thử đời sống biến d(orders):.....	256
- Kiểm thử đời sống biến e(Order):.....	256
- Kiểm thử đời sống biến f(user):.....	257
- Kiểm thử đời sống biến g(success):.....	257
19. Customer - See a single order.....	257
- Kiểm thử đời sống biến a(req):.....	258
- Kiểm thử đời sống biến b(res):.....	259
- Kiểm thử đời sống biến c(next):.....	259
- Kiểm thử đời sống biến d(order):.....	260
- Kiểm thử đời sống biến e(Order):.....	260
- Kiểm thử đời sống biến f(id):.....	261
- Kiểm thử đời sống biến g(user):.....	261
- Kiểm thử đời sống biến h(name email):.....	262
- Kiểm thử đời sống biến i(success):.....	262
20. Admin - See All Users.....	262
- Kiểm thử đời sống biến a(req):.....	263
- Kiểm thử đời sống biến b(res):.....	263
- Kiểm thử đời sống biến c(next):.....	264
- Kiểm thử đời sống biến d(users):.....	264
- Kiểm thử đời sống biến e(User):.....	264
- Kiểm thử đời sống biến f(success):.....	265
21. Admin - Update User Role.....	265
- Kiểm thử đời sống biến a(req):.....	266
- Kiểm thử đời sống biến b(res):.....	266
- Kiểm thử đời sống biến c(next):.....	267
- Kiểm thử đời sống biến d(newUserData):.....	267
- Kiểm thử đời sống biến e(name):.....	267
- Kiểm thử đời sống biến f(email):.....	268
- Kiểm thử đời sống biến g(role):.....	268
- Kiểm thử đời sống biến h(User):.....	268
- Kiểm thử đời sống biến i(id):.....	269
- Kiểm thử đời sống biến k(new):.....	269
- Kiểm thử đời sống biến l(runValidators):.....	269

- Kiểm thử đời sống biến m(useFindAndModify):.....	270
- Kiểm thử đời sống biến n(success):.....	270
22. Admin - Delete Users.....	270
- Kiểm thử đời sống biến a(req):.....	272
- Kiểm thử đời sống biến b(res):.....	272
- Kiểm thử đời sống biến c(next):.....	273
- Kiểm thử đời sống biến d(user):.....	273
- Kiểm thử đời sống biến e(User):.....	274
- Kiểm thử đời sống biến f(id):.....	274
- Kiểm thử đời sống biến g(imageId):.....	275
- Kiểm thử đời sống biến h(avatar):.....	275
- Kiểm thử đời sống biến i(success):.....	276
- Kiểm thử đời sống biến k(message):.....	276
23. Admin - See All Products.....	277
- Kiểm thử đời sống biến a(req):.....	277
- Kiểm thử đời sống biến b(res):.....	278
- Kiểm thử đời sống biến c(next):.....	278
- Kiểm thử đời sống biến d(products):.....	278
- Kiểm thử đời sống biến e(Product):.....	279
- Kiểm thử đời sống biến f(success):.....	279
24. Admin - Create Products.....	279
- Kiểm thử đời sống biến a(req):.....	281
- Kiểm thử đời sống biến b(res):.....	281
- Kiểm thử đời sống biến c(next):.....	282
- Kiểm thử đời sống biến d(images):.....	282
- Kiểm thử đời sống biến e(imageLinks):.....	283
- Kiểm thử đời sống biến f(i):.....	283
- Kiểm thử đời sống biến g(result):.....	284
- Kiểm thử đời sống biến h(user):.....	284
- Kiểm thử đời sống biến i(product):.....	285
- Kiểm thử đời sống biến k(Product):.....	285
- Kiểm thử đời sống biến l(success):.....	286
25. Admin - Update Products.....	286
- Kiểm thử đời sống biến a(req):.....	288
- Kiểm thử đời sống biến b(res):.....	289
- Kiểm thử đời sống biến c(next):.....	289
- Kiểm thử đời sống biến d(product):.....	290

- Kiểm thử đời sống biến e(Product):.....	290
- Kiểm thử đời sống biến f(id):.....	291
- Kiểm thử đời sống biến g(images):.....	292
- Kiểm thử đời sống biến h(i):.....	292
- Kiểm thử đời sống biến i(imagesLink):.....	293
- Kiểm thử đời sống biến k(result):.....	294
- Kiểm thử đời sống biến l(new):.....	294
- Kiểm thử đời sống biến m(runValidators):.....	295
- Kiểm thử đời sống biến n(useFindAndModify):.....	295
- Kiểm thử đời sống biến o(success):.....	296
26. Admin - Delete Products.....	296
- Kiểm thử đời sống biến a(req):.....	297
- Kiểm thử đời sống biến b(res):.....	298
- Kiểm thử đời sống biến c(next):.....	298
- Kiểm thử đời sống biến d(product):.....	299
- Kiểm thử đời sống biến e(Product):.....	299
- Kiểm thử đời sống biến f(id):.....	300
- Kiểm thử đời sống biến g(success):.....	300
- Kiểm thử đời sống biến h(message):.....	301
- Kiểm thử đời sống biến i(images):.....	301
- Kiểm thử đời sống biến k(i):.....	302
27. Admin - See All Orders on system.....	302
- Kiểm thử đời sống biến a(req):.....	303
- Kiểm thử đời sống biến b(res):.....	303
- Kiểm thử đời sống biến c(next):.....	304
- Kiểm thử đời sống biến d(orders):.....	304
- Kiểm thử đời sống biến e(Order):.....	304
- Kiểm thử đời sống biến f(totalAmount):.....	305
- Kiểm thử đời sống biến g(totalPrice):.....	305
- Kiểm thử đời sống biến h(success):.....	305
28. Admin - Update An Order.....	306
- Kiểm thử đời sống biến a(req):.....	307
- Kiểm thử đời sống biến b(res):.....	307
- Kiểm thử đời sống biến c(next):.....	308
- Kiểm thử đời sống biến d(order):.....	308
- Kiểm thử đời sống biến e(Order):.....	308
- Kiểm thử đời sống biến f(id):.....	309

- Kiểm thử đời sống biến g(orderStatus):.....	309
- Kiểm thử đời sống biến h(orderItems):.....	309
- Kiểm thử đời sống biến i(product):.....	310
- Kiểm thử đời sống biến k(o):.....	310
- Kiểm thử đời sống biến l(quantity):.....	311
- Kiểm thử đời sống biến m(status):.....	311
- Kiểm thử đời sống biến n(deliveredAt):.....	312
- Kiểm thử đời sống biến o(success):.....	312
29. Admin - Delete An Order.....	312
- Kiểm thử đời sống biến a(req):.....	313
- Kiểm thử đời sống biến b(res):.....	314
- Kiểm thử đời sống biến c(next):.....	314
- Kiểm thử đời sống biến d(order):.....	315
- Kiểm thử đời sống biến e(Order):.....	315
- Kiểm thử đời sống biến f(id):.....	316
- Kiểm thử đời sống biến g(success):.....	316
30. Admin - See All Reviews of a Product.....	316
- Kiểm thử đời sống biến a(req):.....	317
- Kiểm thử đời sống biến b(res):.....	317
- Kiểm thử đời sống biến c(next):.....	318
- Kiểm thử đời sống biến d(product):.....	318
- Kiểm thử đời sống biến e(Product):.....	318
- Kiểm thử đời sống biến f(id):.....	319
- Kiểm thử đời sống biến g(success):.....	319
- Kiểm thử đời sống biến h(reviews):.....	319
31. Admin - Delete A Product Review.....	320
- Kiểm thử đời sống biến a(req):.....	321
- Kiểm thử đời sống biến b(res):.....	322
- Kiểm thử đời sống biến c(next):.....	322
- Kiểm thử đời sống biến d(product):.....	323
- Kiểm thử đời sống biến e(Product):.....	323
- Kiểm thử đời sống biến f(productId):.....	324
- Kiểm thử đời sống biến g(reviews):.....	324
- Kiểm thử đời sống biến h(rev):.....	325
- Kiểm thử đời sống biến i(id):.....	325
- Kiểm thử đời sống biến k(avg):.....	326
- Kiểm thử đời sống biến l(ratings):.....	326

- Kiểm thử đời sống biển m(<i>numOfReviews</i>):.....	327
- Kiểm thử đời sống biển n(<i>new</i>):.....	327
- Kiểm thử đời sống biển o(<i>runValidators</i>):.....	328
- Kiểm thử đời sống biển p(<i>useFindAndModify</i>):.....	328
- Kiểm thử đời sống biển r(<i>success</i>):.....	329
CHƯƠNG VII: BUGS REPORT.....	330
1. Report 01.....	330
2. Report 02.....	331
3. Report 03.....	333
4. Report 04.....	335
5. Report 05.....	338
TÀI LIỆU THAM KHẢO.....	341

DANH SÁCH CÁC BẢNG

Table 1: Bảng phân công công việc

Table 2: Bảng Giao Công Việc Test

Table 3: Tiêu chí kiểm thử

Table 4: Mức độ nghiêm trọng của lỗi

Table 5: Số liệu kiểm tra sản phẩm

Table 6: Đặc tả usecase đăng ký

Table 7: Đặc tả usecase đăng nhập

Table 8: Đặc tả usecase quên mật khẩu

Table 9: Đặc tả usecase cài lại mật khẩu

Table 9: Đặc tả usecase Xem tất cả các sản phẩm

Table 10: Đặc tả usecase Xem chi tiết thông tin 1 sản phẩm

Table 11: Đặc tả usecase Liên hệ nhóm để tài thông qua email

Table 12: Đặc tả usecase Tìm kiếm sản phẩm theo từ khóa

Table 13: Đặc tả usecase Lọc sản phẩm

Table 14: Đặc tả usecase Đăng xuất

Table 15: Đặc tả usecase Xem thông tin cá nhân

Table 16: Đặc tả usecase Cập nhật thông tin cá nhân

Table 16: Đặc tả usecase Cập nhật mật khẩu

Table 17: Đặc tả usecase Tạo đánh giá

Table 18: Đặc tả usecase Thêm sản phẩm vào Giỏ hàng

Table 19: Đặc tả usecase Đặt hàng

Table 20: Đặc tả usecase Thanh toán

Table 21: Đặc tả usecase Xem tất cả các đơn đặt hàng

Table 22: Đặc tả usecase Xem thông tin chi tiết 1 đơn đặt hàng

Table 23: Đặc tả usecase Xem tất cả các người dùng trên hệ thống

Table 24: *Đặc tả usecase Nâng cấp quyền hạn người dùng*

Table 25: *Đặc tả usecase Xóa người dùng*

Table 26: *Đặc tả usecase Xem tất cả các sản phẩm có trên hệ thống*

Table 27: *Đặc tả usecase Tạo mới 1 sản phẩm*

Table 28: *Đặc tả usecase Cập nhật 1 sản phẩm*

Table 29: *Đặc tả usecase Xóa 1 sản phẩm*

Table 30: *Đặc tả usecase Xem các đơn hàng trên hệ thống*

Table 31: *Đặc tả usecase Cập nhật trạng thái đơn hàng trên hệ thống*

Table 32: *Đặc tả usecase Xóa đơn hàng trên hệ thống*

Table 33: *Đặc tả usecase Xem tất cả các đánh giá của 1 sản phẩm trên hệ thống*

Table 34: *Đặc tả usecase Xóa các đánh giá của 1 sản phẩm trên hệ thống*

Table 5.1: *Code Backend Đăng ký tài khoản*

Table 5.2: *Code Backend Đăng nhập tài khoản*

Table 5.3: *Code Backend Quên mật khẩu*

Table 5.4: *Code Backend Cài lại mật khẩu*

Table 5.5: *Code Backend Xem tất cả các sản phẩm*

Table 5.6: *Code Backend Xem thông tin chi tiết 1 sản phẩm*

Table 5.10: *Code Backend Đăng xuất tài khoản*

Table 5.11: *Code Backend Xem thông tin chi tiết tài khoản*

Table 5.12: *Code Backend Cập nhật thông tin tài khoản*

Table 5.13: *Code Backend Cập nhật mật khẩu tài khoản*

Table 5.14: *Code Backend Đánh giá 1 sản phẩm*

Table 5.16: *Code Backend Đặt đơn hàng*

Table 5.17: *Code Backend Thanh toán đơn hàng*

Table 5.18: *Code Backend Xem tất cả các đơn hàng*

Table 5.19: *Code Backend Xem thông tin 1 đơn hàng*

Table 5.20: Code Backend Xem tất cả các người dùng

Table 5.21: Code Backend Nâng cấp quyền hạn người dùng

Table 5.22: Code Backend Xóa người dùng

Table 5.23: Code Backend Xem tất cả các sản phẩm

Table 5.24: Code Backend Tạo sản phẩm mới

Table 5.25: Code Backend Cập nhật 1 sản phẩm

Table 5.26: Code Backend Xóa sản phẩm

Table 5.27: Code Backend Xem tất cả các đơn hàng trên hệ thống

Table 5.28: Code Backend Cập nhật trạng thái đơn hàng

Table 5.29: Code Backend Xóa đơn hàng

Table 5.30: Code Backend Xem tất cả các đánh giá của 1 sản phẩm

Table 5.31: Code Backend Xóa đánh giá

Table 6.1: Code Backend Đăng ký tài khoản

Table 6.2: Code Backend Đăng nhập tài khoản

Table 6.3: Code Backend Quên mật khẩu

Table 6.4: Code Backend Cài lại mật khẩu

Table 6.5: Code Backend Xem tất cả các sản phẩm

Table 6.6: Code Backend Xem thông tin chi tiết 1 sản phẩm

Table 6.10: Code Backend Đăng xuất tài khoản

Table 6.11: Code Backend Xem thông tin chi tiết tài khoản

Table 6.12: Code Backend Cập nhật thông tin tài khoản

Table 6.13: Code Backend Cập nhật mật khẩu tài khoản

Table 6.14: Code Backend Đánh giá 1 sản phẩm

Table 6.16: Code Backend Đặt đơn hàng

Table 6.17: Code Backend Thanh toán đơn hàng

Table 6.18: Code Backend Xem tất cả các đơn hàng

Table 6.19: Code Backend Xem thông tin 1 đơn hàng

Table 6.20: Code Backend Xem tất cả các người dùng

Table 6.21: Code Backend Nâng cấp quyền hạn người dùng

Table 6.22: Code Backend Xóa người dùng

Table 6.23: Code Backend Xem tất cả các sản phẩm

Table 6.24: Code Backend Tạo sản phẩm mới

Table 6.25: Code Backend Cập nhật 1 sản phẩm

Table 6.26: Code Backend Xóa sản phẩm

Table 6.27: Code Backend Xem tất cả các đơn hàng trên hệ thống

Table 6.28: Code Backend Cập nhật trạng thái đơn hàng

Table 6.29: Code Backend Xóa đơn hàng

Table 6.30: Code Backend Xem tất cả các đánh giá của 1 sản phẩm

Table 6.31: Code Backend Xóa đánh giá

Table 7.1: Bảng Report 01

Table 7.2: Bảng Report 02

Table 7.3: Bảng Report 03

Table 7.4: Bảng Report 04

Table 7.5: Bảng Report 05

DANH SÁCH CÁC ẢNH

Hình 1: Sơ đồ usecase tổng

Hình 2: Usecases của Actor User

Hình 3: Usecases của Actor Customer

Hình 4: Usecases của Actor Admin

Hình 4.1.1 Giao diện trang chủ

Hình 4.2.1 Giao diện trang đăng nhập, đăng ký

Hình 4.3.1 Giao diện trang sản phẩm theo danh mục sản phẩm

Hình 4.4.1 Giao diện trang chi tiết sản phẩm

Hình 4.4.2 Giao diện trang đánh giá, review

Hình 4.5.1 Giao diện header

Hình 4.2.5.2 Giao diện giỏ hàng

Hình 4.6.1 Giao diện trang điền thông tin người mua hàng

Hình 4.6.2 Giao diện trang thanh toán

Hình 4.6.3 Giao diện trang thanh toán

Hình 4.6.4 Giao diện trang xác nhận thanh toán thành công

Hình 4.7.1 Giao diện tìm kiếm

Hình 4.8.1 Giao diện lọc theo giá

Hình 4.8.2 Giao diện lọc theo review

Hình 4.9 Giao diện trang liên hệ

Hình 4.10.1. Giao diện chính trang quản trị viên

Hình 4.10.2. Giao diện danh sách sản phẩm trên hệ thống được quản lý bởi quản trị viên

Hình 4.10.3. Giao diện thêm sản phẩm mới được quản lý bởi quản trị viên

Hình 4.10.4. Giao diện cập nhật sản phẩm được quản lý bởi quản trị viên

Hình 4.10.5 Giao diện danh sách đơn đặt hàng trên hệ thống được quản lý bởi quản trị viên

Hình 4.10.6 . Giao diện chỉnh sửa trạng thái đơn đặt hàng được quản lý bởi quản trị viên

Hình 4.10.7. Giao diện danh sách users trên hệ thống được quản lý bởi admin

Hình 4.10.8. Giao diện nâng cấp quyền hạn tài khoản user được quản lý bởi quản trị viên

Hình 4.10.9 Giao diện các đánh giá sản phẩm bởi users được quản lý bởi quản trị viên

Hình 4.10.10. Giao diện danh sách các đánh giá 1 sản phẩm bởi users được quản lý bởi quản trị viên

Hình 5.1: Đồ thị dòng điều khiển Đăng ký tài khoản

Hình 5.2: Đồ thị dòng điều khiển Đăng nhập tài khoản

Hình 5.3: Đồ thị dòng điều khiển Quên mật khẩu

Hình 5.4: Đồ thị dòng điều khiển Cài lại mật khẩu

Hình 5.5: Đồ thị dòng điều khiển Xem tất cả các sản phẩm

Hình 5.6: Đồ thị dòng điều khiển Xem thông tin chi tiết 1 sản phẩm

Hình 5.10: Đồ thị dòng điều khiển Đăng xuất tài khoản

Hình 5.11: Đồ thị dòng điều khiển Xem thông tin chi tiết tài khoản

Hình 5.12: Đồ thị dòng điều khiển Cập nhật thông tin tài khoản

Hình 5.13: Đồ thị dòng điều khiển Cập nhật mật khẩu tài khoản

Hình 5.14: Đồ thị dòng điều khiển Đánh giá 1 sản phẩm

Hình 5.16: Đồ thị dòng điều khiển Đặt đơn hàng

Hình 5.17: Đồ thị dòng điều khiển Thanh toán đơn hàng

Hình 5.18: Đồ thị dòng điều khiển Xem tất cả các đơn hàng

Hình 5.19: Đồ thị dòng điều khiển Xem thông tin 1 đơn hàng

Hình 5.20: Đồ thị dòng điều khiển Xem tất cả các người dùng

Hình 5.21: Đồ thị dòng điều khiển Nâng cấp quyền hạn người dùng

Hình 5.22: Đồ thị dòng điều khiển Xóa người dùng

Hình 5.23: Đồ thị dòng điều khiển Xem tất cả các sản phẩm

Hình 5.24: Đồ thị dòng điều khiển Tạo sản phẩm mới

Hình 5.25: Đồ thị dòng điều khiển Cập nhật 1 sản phẩm

Hình 5.26: Đồ thị dòng điều khiển Xóa sản phẩm

Hình 5.27: Đồ thị dòng điều khiển Xem tất cả các đơn hàng trên hệ thống

Hình 5.28: Đồ thị dòng điều khiển Cập nhật trạng thái đơn hàng

Hình 5.29: Đồ thị dòng điều khiển Xóa đơn hàng

Hình 5.30: Đồ thị dòng điều khiển Xem tất cả các đánh giá của 1 sản phẩm

Hình 5.31: Đồ thị dòng điều khiển Xóa đánh giá

Hình 6.1: Đồ thị dòng dữ liệu Đăng ký tài khoản

Hình 6.1.a: Đồ thị đòi hỏi biến a(req) Đăng ký tài khoản

Hình 6.1.b: Đồ thị đòi hỏi biến b(res) Đăng ký tài khoản

Hình 6.1.c: Đồ thị đòi hỏi biến c(next) Đăng ký tài khoản

Hình 6.1.d: Đồ thị đòi hỏi biến d(myCloud) Đăng ký tài khoản

Hình 6.1.e: Đồ thị đòi hỏi biến e/avatar) Đăng ký tài khoản

Hình 6.1.f: Đồ thị đòi hỏi biến f(name) Đăng ký tài khoản

Hình 6.1.g: Đồ thị đòi hỏi biến g(email) Đăng ký tài khoản

Hình 6.1.h: Đồ thị đòi hỏi biến h(password) Đăng ký tài khoản

Hình 6.1.i: Đồ thị đòi hỏi biến i(user) Đăng ký tài khoản

Hình 6.2: Đồ thị dòng dữ liệu Đăng nhập tài khoản

Hình 6.2.a: Đồ thị đòi hỏi biến a(req) Đăng nhập tài khoản

Hình 6.2.b: Đồ thị đòi hỏi biến b(res) Đăng nhập tài khoản

Hình 6.2.c: Đồ thị đòi hỏi biến c(next) Đăng nhập tài khoản

Hình 6.2.d: Đồ thị đòi hỏi biến d(email) Đăng nhập tài khoản

Hình 6.2.e: Đồ thị đòi hỏi biến e(password) Đăng nhập tài khoản

Hình 6.2.f: Đồ thị đòi hỏi biến f(user) Đăng nhập tài khoản

Hình 6.2.g: Đồ thị đòi hỏi biến g(isPasswordMatched) Đăng ký tài khoản

Hình 6.3: Đồ thị dòng dữ liệu Quên mật khẩu

Hình 6.3.a: Đồ thị đòi hỏi biến a(req) Quên mật khẩu

Hình 6.3.b: Đồ thị đòi hỏi biến b(res) Quên mật khẩu

Hình 6.3.c: Đồ thị đòi hỏi biến c(next) Quên mật khẩu

Hình 6.3.d: Đồ thị đòi hỏi biến d(user) Quên mật khẩu

Hình 6.3.e: Đồ thị đòi hỏi biến e(email) Quên mật khẩu

Hình 6.3.f: Đồ thị đòi hỏi biến f(resetToken) Quên mật khẩu

Hình 6.3.g: Đồ thị đòi hỏi biến g(resetPasswordUrl) Quên mật khẩu

Hình 6.3.h: Đồ thị đòi hỏi biến h(message) Quên mật khẩu

Hình 6.3.i: Đồ thị đòi hỏi biến i(subject) Quên mật khẩu

Hình 6.3.k: Đồ thị đòi hỏi biến k(error) Quên mật khẩu

Hình 6.3.l: Đồ thị đòi hỏi biến l(resetPasswordToken) Quên mật khẩu

Hình 6.3.m: Đồ thị đòi hỏi biến m(resetPasswordExpire) Quên mật khẩu

Hình 6.4: Đồ thị dòng dữ liệu Cài lại mật khẩu

Hình 6.4.a: Đồ thị đòi hỏi biến a(req) Cài lại mật khẩu

Hình 6.4.b: Đồ thị đòi hỏi biến b(res) Cài lại mật khẩu

Hình 6.4.c: Đồ thị đòi hỏi biến c(next) Cài lại mật khẩu

Hình 6.4.d: Đồ thị đòi hỏi biến d(resetPasswordToken) Cài lại mật khẩu

Hình 6.4.e: Đồ thị đòi hỏi biến e(token) Cài lại mật khẩu

Hình 6.4.f: Đồ thị đòi hỏi biến f(user) Cài lại mật khẩu

Hình 6.4.g: Đồ thị đòi hỏi biến g(resetPasswordExpire) Cài lại mật khẩu

Hình 6.4.h: Đồ thị đòi hỏi biến h(confirmPassword) Cài lại mật khẩu

Hình 6.4.i: Đồ thị đòi hỏi biến i(password) Cài lại mật khẩu

Hình 6.5: Đồ thị dòng dữ liệu Xem tất cả các sản phẩm

Hình 6.5.a: Đồ thị đòi hỏi biến a(req) Xem tất cả sản phẩm

Hình 6.5.b: Đồ thị đòi hỏi biến b(res) Xem tất cả sản phẩm

Hình 6.5.c: Đồ thị đòi hỏi biến c(next) Xem tất cả sản phẩm

Hình 6.5.d: Đò thị đòn sóng biến d(resultPerPage) Xem tất cả sản phẩm

Hình 6.5.e: Đò thị đòn sóng biến e(productsCount) Xem tất cả sản phẩm

Hình 6.5.f: Đò thị đòn sóng biến f(Product) Xem tất cả sản phẩm

Hình 6.5.g: Đò thị đòn sóng biến g(apiFeature) Xem tất cả sản phẩm

Hình 6.5.h: Đò thị đòn sóng biến h(products) Xem tất cả sản phẩm

Hình 6.5.i: Đò thị đòn sóng biến i(filteredProductsCount) Xem tất cả sản phẩm

Hình 6.5.k: Đò thị đòn sóng biến k(success) Xem tất cả sản phẩm

Hình 6.6: Đò thị dòng dữ liệu Xem thông tin chi tiết 1 sản phẩm

Hình 6.6.a: Đò thị đòn sóng biến a(req) Xem thông tin chi tiết 1 sản phẩm

Hình 6.6.b: Đò thị đòn sóng biến b(res) Xem thông tin chi tiết 1 sản phẩm

Hình 6.6.c: Đò thị đòn sóng biến c(next) Xem thông tin chi tiết 1 sản phẩm

Hình 6.6.d: Đò thị đòn sóng biến d(product) Xem thông tin chi tiết 1 sản phẩm

Hình 6.6.e: Đò thị đòn sóng biến e(Product) Xem thông tin chi tiết 1 sản phẩm

Hình 6.6.f: Đò thị đòn sóng biến f(id) Xem thông tin chi tiết 1 sản phẩm

Hình 6.6.g: Đò thị đòn sóng biến g(success) Xem thông tin chi tiết 1 sản phẩm

Hình 6.10: Đò thị dòng dữ liệu Đăng xuất tài khoản

Hình 6.10.a: Đò thị đòn sóng biến a(req) Đăng xuất tài khoản

Hình 6.10.b: Đò thị đòn sóng biến b(res) Đăng xuất tài khoản

Hình 6.10.c: Đò thị đòn sóng biến c(next) Đăng xuất tài khoản

Hình 6.10.d: Đò thị đòn sóng biến d(expires) Đăng xuất tài khoản

Hình 6.10.e: Đò thị đòn sóng biến e(httpOnly) Đăng xuất tài khoản

Hình 6.10.f: Đò thị đòn sóng biến f(success) Đăng xuất tài khoản

Hình 6.10.g: Đò thị đòn sóng biến g(message) Đăng xuất tài khoản

Hình 6.11: Đò thị dòng dữ liệu Xem thông tin chi tiết tài khoản

Hình 6.11.a: Đò thị đòn sóng biến a(req) Xem thông tin chi tiết tài khoản

Hình 6.11.b: Đò thị đòn sóng biến b(res) Xem thông tin chi tiết tài khoản

Hình 6.11.c: Đò thị đòi sóng biến c(next) Xem thông tin chi tiết tài khoản

Hình 6.11.d: Đò thị đòi sóng biến d(user) Xem thông tin chi tiết tài khoản

Hình 6.11.e: Đò thị đòi sóng biến e(User) Xem thông tin chi tiết tài khoản

Hình 6.11.f: Đò thị đòi sóng biến f(id) Xem thông tin chi tiết tài khoản

Hình 6.11.g: Đò thị đòi sóng biến g(success) Xem thông tin chi tiết tài khoản

Hình 6.12: Đò thị dòng dữ liệu Cập nhật thông tin tài khoản

Hình 6.12.a: Đò thị đòi sóng biến a(req) Cập nhật thông tin tài khoản

Hình 6.12.b: Đò thị đòi sóng biến b(res) Cập nhật thông tin tài khoản

Hình 6.12.c: Đò thị đòi sóng biến c(next) Cập nhật thông tin tài khoản

Hình 6.12.d: Đò thị đòi sóng biến d(newUserData) Cập nhật thông tin tài khoản

Hình 6.12.e: Đò thị đòi sóng biến e(name) Cập nhật thông tin tài khoản

Hình 6.12.f: Đò thị đòi sóng biến f(email) Cập nhật thông tin tài khoản

Hình 6.12.g: Đò thị đòi sóng biến g(avatar) Cập nhật thông tin tài khoản

Hình 6.12.h: Đò thị đòi sóng biến h(user) Cập nhật thông tin tài khoản

Hình 6.12.i: Đò thị đòi sóng biến i(User) Cập nhật thông tin tài khoản

Hình 6.12.j: Đò thị đòi sóng biến k(imageId) Cập nhật thông tin tài khoản

Hình 6.12.l: Đò thị đòi sóng biến l(myCloud) Cập nhật thông tin tài khoản

Hình 6.12.m: Đò thị đòi sóng biến m(success) Cập nhật thông tin tài khoản

Hình 6.13: Đò thị dòng dữ liệu Cập nhật mật khẩu tài khoản

Hình 6.13.a: Đò thị đòi sóng biến a(req) Cập nhật mật khẩu tài khoản

Hình 6.13.b: Đò thị đòi sóng biến b(res) Cập nhật mật khẩu tài khoản

Hình 6.13.c: Đò thị đòi sóng biến c(next) Cập nhật mật khẩu tài khoản

Hình 6.13.d: Đò thị đòi sóng biến d(user) Cập nhật mật khẩu tài khoản

Hình 6.13.e: Đò thị đòi sóng biến e(User) Cập nhật mật khẩu tài khoản

Hình 6.13.f: Đò thị đòi sóng biến f(id) Cập nhật mật khẩu tài khoản

Hình 6.13.g: Đò thị đòi sóng biến g(password) Cập nhật mật khẩu tài khoản

Hình 6.13.h: *Đồ thị đòi hỏi biến h(isPasswordMatched) Cập nhật mật khẩu tài khoản*

Hình 6.13.i: *Đồ thị đòi hỏi biến i(oldPassword) Cập nhật mật khẩu tài khoản*

Hình 6.13.k: *Đồ thị đòi hỏi biến k(newPassword) Cập nhật mật khẩu tài khoản*

Hình 6.13.l: *Đồ thị đòi hỏi biến l(confirmPassword) Cập nhật mật khẩu tài khoản*

Hình 6.14: *Đồ thị dòng dữ liệu Dánh giá 1 sản phẩm*

Hình 6.14.a: *Đồ thị đòi hỏi biến a(req) Dánh giá 1 sản phẩm*

Hình 6.14.b: *Đồ thị đòi hỏi biến b(res) Dánh giá 1 sản phẩm*

Hình 6.14.c: *Đồ thị đòi hỏi biến c(next) Dánh giá 1 sản phẩm*

Hình 6.14.d: *Đồ thị đòi hỏi biến d(rating) Dánh giá 1 sản phẩm*

Hình 6.14.e: *Đồ thị đòi hỏi biến e(comment) Dánh giá 1 sản phẩm*

Hình 6.14.f: *Đồ thị đòi hỏi biến f(productId) Dánh giá 1 sản phẩm*

Hình 6.14.g: *Đồ thị đòi hỏi biến g(review) Dánh giá 1 sản phẩm*

Hình 6.14.h: *Đồ thị đòi hỏi biến h(user) Dánh giá 1 sản phẩm*

Hình 6.14.i: *Đồ thị đòi hỏi biến i(name) Dánh giá 1 sản phẩm*

Hình 6.14.k: *Đồ thị đòi hỏi biến k(product) Dánh giá 1 sản phẩm*

Hình 6.14.l: *Đồ thị đòi hỏi biến l(Product) Dánh giá 1 sản phẩm*

Hình 6.14.m: *Đồ thị đòi hỏi biến m(isReviewed) Dánh giá 1 sản phẩm*

Hình 6.14.n: *Đồ thị đòi hỏi biến n(reviews) Dánh giá 1 sản phẩm*

Hình 6.14.o: *Đồ thị đòi hỏi biến o(rev) Dánh giá 1 sản phẩm*

Hình 6.14.p: *Đồ thị đòi hỏi biến p(avg) Dánh giá 1 sản phẩm*

Hình 6.14.r: *Đồ thị đòi hỏi biến r(ratings) Dánh giá 1 sản phẩm*

Hình 6.14.s: *Đồ thị đòi hỏi biến s(success) Dánh giá 1 sản phẩm*

Hình 6.16: *Đồ thị dòng dữ liệu Đặt đơn hàng*

Hình 6.16.a: *Đồ thị đòi hỏi biến a(req) Đặt đơn hàng*

Hình 6.16.b: *Đồ thị đòi hỏi biến b(res) Đặt đơn hàng*

Hình 6.16.c: *Đồ thị đòi hỏi biến c(next) Đặt đơn hàng*

Hình 6.16.d: *Đồ thị đời sống biến d(shippingInfo) Đặt đơn hàng*

Hình 6.16.e: *Đồ thị đời sống biến e(orderItems) Đặt đơn hàng*

Hình 6.16.f: *Đồ thị đời sống biến f(paymentInfo) Đặt đơn hàng*

Hình 6.16.g: *Đồ thị đời sống biến g(itemsPrice) Đặt đơn hàng*

Hình 6.16.h: *Đồ thị đời sống biến h(taxPrice) Đặt đơn hàng*

Hình 6.16.i: *Đồ thị đời sống biến i(shippingPrice) Đặt đơn hàng*

Hình 6.16.k: *Đồ thị đời sống biến k(totalPrice) Đặt đơn hàng*

Hình 6.16.l: *Đồ thị đời sống biến l(order) Đặt đơn hàng*

Hình 6.1.m: *Đồ thị đời sống biến m(Order) Đặt đơn hàng*

Hình 6.16.n: *Đồ thị đời sống biến n(paidAt) Đặt đơn hàng*

Hình 6.16.o: *Đồ thị đời sống biến o(user) Đặt đơn hàng*

Hình 6.16.p: *Đồ thị đời sống biến p(success) Đặt đơn hàng*

Hình 6.17: *Đồ thị dòng dữ liệu Thanh toán đơn hàng*

Hình 6.17.a: *Đồ thị đời sống biến a(req) Thanh toán đơn hàng*

Hình 6.17.b: *Đồ thị đời sống biến b(res) Thanh toán đơn hàng*

Hình 6.17.c: *Đồ thị đời sống biến c(next) Thanh toán đơn hàng*

Hình 6.17.d: *Đồ thị đời sống biến d(myPayment) Thanh toán đơn hàng*

Hình 6.17.e: *Đồ thị đời sống biến e(stripe) Thanh toán đơn hàng*

Hình 6.17.f: *Đồ thị đời sống biến f(paymentIntents) Thanh toán đơn hàng*

Hình 6.17.g: *Đồ thị đời sống biến g(amount) Thanh toán đơn hàng*

Hình 6.17.h: *Đồ thị đời sống biến h(currency) Thanh toán đơn hàng*

Hình 6.17.i: *Đồ thị đời sống biến i(metadata) Thanh toán đơn hàng*

Hình 6.17.k: *Đồ thị đời sống biến k(company) Thanh toán đơn hàng*

Hình 6.17.l: *Đồ thị đời sống biến l(success) Thanh toán đơn hàng*

Hình 6.18: *Đồ thị dòng dữ liệu Xem tất cả các đơn hàng*

Hình 6.18.a: *Đồ thị đời sống biến a(req) Xem tất cả các đơn hàng*

Hình 6.18.b: Đồ thị đời sóng biến b(res) Xem tất cả các đơn hàng

Hình 6.18.c: Đồ thị đời sóng biến c(next) Xem tất cả các đơn hàng

Hình 6.18.d: Đồ thị đời sóng biến d(orders) Xem tất cả các đơn hàng

Hình 6.18.e: Đồ thị đời sóng biến e(Order) Xem tất cả các đơn hàng

Hình 6.18.f: Đồ thị đời sóng biến f(user) Xem tất cả các đơn hàng

Hình 6.18.g: Đồ thị đời sóng biến g(success) Xem tất cả các đơn hàng

Hình 6.19: Đồ thị dòng dữ liệu Xem thông tin 1 đơn hàng

Hình 6.19.a: Đồ thị đời sóng biến a(req) Xem thông tin 1 đơn hàng

Hình 6.19.b: Đồ thị đời sóng biến b(res) Xem thông tin 1 đơn hàng

Hình 6.19.c: Đồ thị đời sóng biến c(next) Xem thông tin 1 đơn hàng

Hình 6.19.d: Đồ thị đời sóng biến d(order) Xem thông tin 1 đơn hàng

Hình 6.19.e: Đồ thị đời sóng biến e(Order) Xem thông tin 1 đơn hàng

Hình 6.19.f: Đồ thị đời sóng biến f(id) Xem thông tin 1 đơn hàng

Hình 6.19.g: Đồ thị đời sóng biến g(user) Xem thông tin 1 đơn hàng

Hình 6.19.h: Đồ thị đời sóng biến h(name email) Xem thông tin 1 đơn hàng

Hình 6.19.i: Đồ thị đời sóng biến ai(success) Xem thông tin 1 đơn hàng

Hình 6.20: Đồ thị dòng dữ liệu Xem tất cả các người dùng

Hình 6.20.a: Đồ thị đời sóng biến a(req) Xem tất cả các người dùng

Hình 6.20.b: Đồ thị đời sóng biến b(res) Xem tất cả các người dùng

Hình 6.20.c: Đồ thị đời sóng biến c(next) Xem tất cả các người dùng

Hình 6.20.d: Đồ thị đời sóng biến d(users) Xem tất cả các người dùng

Hình 6.20.e: Đồ thị đời sóng biến e(User) Xem tất cả các người dùng

Hình 6.20.f: Đồ thị đời sóng biến f(success) Xem tất cả các người dùng

Hình 6.21: Đồ thị dòng dữ liệu Nâng cấp quyền hạn người dùng

Hình 6.21.a: Đồ thị đời sóng biến a(req) Đăng ký tài khoản

Hình 6.21.b: Đồ thị đời sóng biến b(res) Đăng ký tài khoản

Hình 6.21.c: Đồ thị đòi hỏi biến c(next) Đăng ký tài khoản

Hình 6.21.d: Đồ thị đòi hỏi biến d(newUserData) Đăng ký tài khoản

Hình 6.21.e: Đồ thị đòi hỏi biến e(name) Đăng ký tài khoản

Hình 6.21.f: Đồ thị đòi hỏi biến f(email) Đăng ký tài khoản

Hình 6.21.g: Đồ thị đòi hỏi biến g(role) Đăng ký tài khoản

Hình 6.21.h: Đồ thị đòi hỏi biến h(User) Đăng ký tài khoản

Hình 6.21.i: Đồ thị đòi hỏi biến i(id) Đăng ký tài khoản

Hình 6.21.j: Đồ thị đòi hỏi biến k(new) Đăng ký tài khoản

Hình 6.21.l: Đồ thị đòi hỏi biến l(runValidators) Đăng ký tài khoản

Hình 6.21.m: Đồ thị đòi hỏi biến m(useFindAndModify) Đăng ký tài khoản

Hình 6.21.n: Đồ thị đòi hỏi biến n(success) Đăng ký tài khoản

Hình 6.22: Đồ thị dòng dữ liệu Xóa người dùng

Hình 6.22.a: Đồ thị đòi hỏi biến a(req) Nâng cấp quyền hạn người dùng

Hình 6.22.b: Đồ thị đòi hỏi biến b(res) Nâng cấp quyền hạn người dùng

Hình 6.22.c: Đồ thị đòi hỏi biến c(next) Nâng cấp quyền hạn người dùng

Hình 6.22.d: Đồ thị đòi hỏi biến d(user) Nâng cấp quyền hạn người dùng

Hình 6.22.e: Đồ thị đòi hỏi biến e(User) Nâng cấp quyền hạn người dùng

Hình 6.22.f: Đồ thị đòi hỏi biến f(id) Nâng cấp quyền hạn người dùng

Hình 6.22.g: Đồ thị đòi hỏi biến g(imageId) Nâng cấp quyền hạn người dùng

Hình 6.22.h: Đồ thị đòi hỏi biến h(avatar) Nâng cấp quyền hạn người dùng

Hình 6.22.i: Đồ thị đòi hỏi biến i(success) Nâng cấp quyền hạn người dùng

Hình 6.22.k: Đồ thị đòi hỏi biến k(message) Nâng cấp quyền hạn người dùng

Hình 6.23: Đồ thị dòng dữ liệu Xem tất cả các sản phẩm

Hình 6.23.a: Đồ thị đòi hỏi biến a(req) Xem tất cả các sản phẩm

Hình 6.23.b: Đồ thị đòi hỏi biến b(res) Xem tất cả các sản phẩm

Hình 6.23.c: Đồ thị đòi hỏi biến c(next) Xem tất cả các sản phẩm

Hình 6.23.d: *Đồ thị đời sống biển d(products)* Xem tất cả các sản phẩm

Hình 6.23.e: *Đồ thị đời sống biển e(Product)* Xem tất cả các sản phẩm

Hình 6.20.f: *Đồ thị đời sống biển f(success)* Xem tất cả các sản phẩm

Hình 6.24: *Đồ thị dòng dữ liệu Tạo sản phẩm mới*

Hình 6.24.a: *Đồ thị đời sống biển a(req)* Tạo sản phẩm mới

Hình 6.24.b: *Đồ thị đời sống biển b(res)* Tạo sản phẩm mới

Hình 6.24.c: *Đồ thị đời sống biển c(next)* Tạo sản phẩm mới

Hình 6.24.d: *Đồ thị đời sống biển d(images)* Tạo sản phẩm mới

Hình 6.24.e: *Đồ thị đời sống biển e(imageLinks)* Tạo sản phẩm mới

Hình 6.24.f: *Đồ thị đời sống biển f(i)* Tạo sản phẩm mới

Hình 6.24.g: *Đồ thị đời sống biển g(result)* Tạo sản phẩm mới

Hình 6.24.h: *Đồ thị đời sống biển h(user)* Tạo sản phẩm mới

Hình 6.24.i: *Đồ thị đời sống biển i(product)* Tạo sản phẩm mới

Hình 6.24.k: *Đồ thị đời sống biển k(Product)* Tạo sản phẩm mới

Hình 6.24.l: *Đồ thị đời sống biển l(success)* Tạo sản phẩm mới

Hình 6.25: *Đồ thị dòng dữ liệu Cập nhật 1 sản phẩm*

Hình 6.25.a: *Đồ thị đời sống biển a(req)* Cập nhật 1 sản phẩm

Hình 6.25.b: *Đồ thị đời sống biển b(res)* Cập nhật 1 sản phẩm

Hình 6.25.c: *Đồ thị đời sống biển c(next)* Cập nhật 1 sản phẩm

Hình 6.25.d: *Đồ thị đời sống biển d(product)* Cập nhật 1 sản phẩm

Hình 6.25.e: *Đồ thị đời sống biển e(Product)* Cập nhật 1 sản phẩm

Hình 6.25.f: *Đồ thị đời sống biển f(id)* Cập nhật 1 sản phẩm

Hình 6.25.g: *Đồ thị đời sống biển g(images)* Cập nhật 1 sản phẩm

Hình 6.25.h: *Đồ thị đời sống biển h(i)* Cập nhật 1 sản phẩm

Hình 6.25.i: *Đồ thị đời sống biển i(imagesLink)* Cập nhật 1 sản phẩm

Hình 6.25.k: *Đồ thị đời sống biển k(result)* Cập nhật 1 sản phẩm

Hình 6.25.l: Đồ thị đời sóng biến l(new) Cập nhật 1 sản phẩm

Hình 6.25.m: Đồ thị đời sóng biến m(runValidators) Cập nhật 1 sản phẩm

Hình 6.25.m: Đồ thị đời sóng biến n(useFindAndModify) Cập nhật 1 sản phẩm

Hình 6.25.o: Đồ thị đời sóng biến o(success) Cập nhật 1 sản phẩm

Hình 6.26: Đồ thị dòng dữ liệu Xóa sản phẩm

Hình 6.26.a: Đồ thị đời sóng biến a(req) Xóa sản phẩm

Hình 6.26.b: Đồ thị đời sóng biến b(res) Xóa sản phẩm

Hình 6.26.c: Đồ thị đời sóng biến c(next) Xóa sản phẩm

Hình 6.26.d: Đồ thị đời sóng biến d(product) Xóa sản phẩm

Hình 6.26.e: Đồ thị đời sóng biến e(Product) Xóa sản phẩm

Hình 6.26.f: Đồ thị đời sóng biến f(id) Xóa sản phẩm

Hình 6.26.g: Đồ thị đời sóng biến g(success) Xóa sản phẩm

Hình 6.26.h: Đồ thị đời sóng biến h(message) Xóa sản phẩm

Hình 6.26.i: Đồ thị đời sóng biến i(images) Xóa sản phẩm

Hình 6.26.k: Đồ thị đời sóng biến k(i) Xóa sản phẩm

Hình 6.27: Đồ thị dòng dữ liệu Xem tất cả các đơn hàng trên hệ thống

Hình 6.27.a: Đồ thị đời sóng biến a(req) Xem tất cả các đơn hàng trên hệ thống

Hình 6.27.b: Đồ thị đời sóng biến b(res) Xem tất cả các đơn hàng trên hệ thống

Hình 6.27.c: Đồ thị đời sóng biến c(next) Xem tất cả các đơn hàng trên hệ thống

Hình 6.27.d: Đồ thị đời sóng biến d(orders) Xem tất cả các đơn hàng trên hệ thống

Hình 6.27.e: Đồ thị đời sóng biến e(Order) Xem tất cả các đơn hàng trên hệ thống

Hình 6.27.f: Đồ thị đời sóng biến f(totalAmount) Xem tất cả các đơn hàng trên hệ thống

Hình 6.27.g: Đồ thị đời sóng biến g(totalPrice) Xem tất cả các đơn hàng trên hệ thống

Hình 6.27.h: Đồ thị đời sóng biến h(success) Xem tất cả các đơn hàng trên hệ thống

Hình 6.28: Đồ thị dòng dữ liệu Cập nhật trạng thái đơn hàng

Hình 6.28.a: Đồ thị đời sóng biến a(req) Cập nhật trạng thái đơn hàng

Hình 6.28.b: *Đồ thị đòi hỏi sóng biến b(res) Cập nhật trạng thái đơn hàng*

Hình 6.28.c: *Đồ thị đòi hỏi sóng biến c(next) Cập nhật trạng thái đơn hàng*

Hình 6.28.d: *Đồ thị đòi hỏi sóng biến d(order) Cập nhật trạng thái đơn hàng*

Hình 6.28.e: *Đồ thị đòi hỏi sóng biến e(Order) Cập nhật trạng thái đơn hàng*

Hình 6.28.f: *Đồ thị đòi hỏi sóng biến f(id) Cập nhật trạng thái đơn hàng*

Hình 6.28.g: *Đồ thị đòi hỏi sóng biến g(orderStatus) Cập nhật trạng thái đơn hàng*

Hình 6.28.h: *Đồ thị đòi hỏi sóng biến h(orderItems) Cập nhật trạng thái đơn hàng*

Hình 6.28.i: *Đồ thị đòi hỏi sóng biến i(product) Cập nhật trạng thái đơn hàng*

Hình 6.28.k: *Đồ thị đòi hỏi sóng biến k(o) Cập nhật trạng thái đơn hàng*

Hình 6.28.l: *Đồ thị đòi hỏi sóng biến l(quantity) Cập nhật trạng thái đơn hàng*

Hình 6.28.m: *Đồ thị đòi hỏi sóng biến m(status) Cập nhật trạng thái đơn hàng*

Hình 6.28.n: *Đồ thị đòi hỏi sóng biến n(deliveredAt) Cập nhật trạng thái đơn hàng*

Hình 6.28.o: *Đồ thị đòi hỏi sóng biến o(success) Cập nhật trạng thái đơn hàng*

Hình 6.29: *Đồ thị dòng dữ liệu Xóa đơn hàng*

Hình 6.29.a: *Đồ thị đòi hỏi sóng biến a(req) Xóa đơn hàng*

Hình 6.29.b: *Đồ thị đòi hỏi sóng biến b(res) Xóa đơn hàng*

Hình 6.29.c: *Đồ thị đòi hỏi sóng biến c(next) Xóa đơn hàng*

Hình 6.29.d: *Đồ thị đòi hỏi sóng biến d(order) Xóa đơn hàng*

Hình 6.29.e: *Đồ thị đòi hỏi sóng biến e(Order) Xóa đơn hàng*

Hình 6.29.f: *Đồ thị đòi hỏi sóng biến f(id) Xóa đơn hàng*

Hình 6.29.g: *Đồ thị đòi hỏi sóng biến g(success) Xóa đơn hàng*

Hình 6.30: *Đồ thị dòng dữ liệu Xem tất cả các đánh giá của 1 sản phẩm*

Hình 6.30.a: *Đồ thị đòi hỏi sóng biến a(req) Xem tất cả các đánh giá của 1 sản phẩm*

Hình 6.30.b: *Đồ thị đòi hỏi sóng biến b(res) Xem tất cả các đánh giá của 1 sản phẩm*

Hình 6.30.c: *Đồ thị đòi hỏi sóng biến c(next) Xem tất cả các đánh giá của 1 sản phẩm*

Hình 6.30.d: *Đồ thị đòi hỏi sóng biến d(product) Xem tất cả các đánh giá của 1 sản phẩm*

Hình 6.30.e: *Đồ thị đời sống biến e(Product)* Xem tất cả các đánh giá của 1 sản phẩm

Hình 6.30.f: *Đồ thị đời sống biến f(id)* Xem tất cả các đánh giá của 1 sản phẩm

Hình 6.30.g: *Đồ thị đời sống biến g(success)* Xem tất cả các đánh giá của 1 sản phẩm

Hình 6.30.h: *Đồ thị đời sống biến h(reviews)* Xem tất cả các đánh giá của 1 sản phẩm

Hình 6.31: *Đồ thị dòng dữ liệu Xóa đánh giá*

Hình 6.31.a: *Đồ thị đời sống biến a(req)* Xóa đánh giá

Hình 6.31.b: *Đồ thị đời sống biến b(res)* Xóa đánh giá

Hình 6.31.c: *Đồ thị đời sống biến c(next)* Xóa đánh giá

Hình 6.31.d: *Đồ thị đời sống biến d(product)* Xóa đánh giá

Hình 6.31.e: *Đồ thị đời sống biến e(Product)* Xóa đánh giá

Hình 6.31.f: *Đồ thị đời sống biến f(productId)* Xóa đánh giá

Hình 6.31.g: *Đồ thị đời sống biến g(reviews)* Xóa đánh giá

Hình 6.31.h: *Đồ thị đời sống biến h(rev)* Xóa đánh giá

Hình 6.31.i: *Đồ thị đời sống biến i(id)* Xóa đánh giá

Hình 6.31.k: *Đồ thị đời sống biến k(avg)* Xóa đánh giá

Hình 6.31.l: *Đồ thị đời sống biến l(ratings)* Xóa đánh giá

Hình 6.31.m: *Đồ thị đời sống biến m(numOfReviews)* Xóa đánh giá

Hình 6.31.n: *Đồ thị đời sống biến n(new)* Xóa đánh giá

Hình 6.31.o: *Đồ thị đời sống biến o(runValidators)* Xóa đánh giá

Hình 6.31.p: *Đồ thị đời sống biến p(useFindAndModify)* Xóa đánh giá

Hình 6.31.r: *Đồ thị đời sống biến r(success)* Xóa đánh giá

PHIẾU NHẬN XÉT CỦA GIÁO VIÊN

Họ và tên Sinh viên 1: Bùi Hà Nhi

MSSV 1: 18110168

Họ và tên Sinh viên 2: Hoàng Minh Quang

MSSV 2: 18110181

Họ và tên Sinh viên 3: Nguyễn Quang Vũ

MSSV 3: 18110241

Ngành: Công nghệ thông tin

Tên đề tài: Kiểm thử website bán sách và thiết bị học tập

Họ và tên Giáo viên hướng dẫn: TS. Nguyễn Trần Thị Văn

Nhận Xét:

1. Về nội dung đề tài & khôi công việc thực hiện:

.....
.....
.....

2. Ưu điểm:

.....
.....
.....

3. Khuyết điểm:

.....
.....
.....

4. Đề nghị cho bảo vệ hay không?

5. Đánh giá loại:

6. Điểm:

Tp. HCM, tháng 12 năm 2020

Giáo viên hướng dẫn

(Ký & ghi rõ họ tên)

TS. Nguyễn Trần Thị Văn

BẢNG PHÂN CÔNG CÔNG VIỆC

Tuần	Công việc	Sản phẩm	Người thực hiện
1 13/10/2021- 20/10/2021	Tìm tài liệu	- 3 ebooks - 6 địa chỉ website	Cả nhóm
	Lập kế hoạch	Project Plan	Cả nhóm
	Phân công công việc	Bảng phân công	Cả nhóm
	Viết báo cáo phần đầu	Bản báo cáo	Cả nhóm
2 20/10/2021- 27/10/2021	Đặc tả yêu cầu	Đặc tả yêu cầu hệ thống	Cả nhóm
	Vẽ Use case	Use case	Bùi Hà Nhi
	Lược đồ chức năng	Lược đồ chức năng	Bùi Hà Nhi
	Mô tả chi tiết các chức năng của hệ thống được xây dựng.	Mô tả chi tiết chức năng	Bùi Hà Nhi, Nguyễn Quang Vũ
	Viết hướng dẫn sử dụng sản phẩm	Hướng dẫn sử dụng website	Nguyễn Quang Vũ
	Viết báo cáo	Bản báo cáo	Cả nhóm
3 27/10/2021-	Test Plan	Test Plan	Cả nhóm

3/11/2021	Phương pháp kiểm thử được áp dụng	Bản báo cáo	Hoàng Minh Quang
4 3/11/2021- 10/11/2021	Kiểm thử hộp trắng Vẽ đồ thị dòng	Đồ thị dòng điều khiển, đồ thị dòng dữ liệu của chức năng, kiểm thử đời sống các biến	Cả nhóm
	Viết báo cáo	Bản báo cáo	Cả nhóm
5 10/11/2021- 17/11/2021	Kiểm thử hộp đen Viết Test Cases	50 Test cases trong file .xlsx	Cả nhóm
	Viết bug report	Bug Reports	Hoàng Minh Quang
6 17/11/2021- 21/11/2021	Hoàn thiện báo cáo	Bản báo cáo hoàn thiện	Cả Nhóm

Table 1: Bảng phân công công việc

CHƯƠNG I: TEST PLAN

1. Tổng quan

1.1. Thông tin dự án

- Tên dự án: Xây dựng website bán sách và thiết bị học tập
- Mã dự án: NewBookstoreFieldProject (NBF)
- Loại sản phẩm: Website
- Ngày bắt đầu: ngày 24 tháng 8 năm 2021
- Ngày kết thúc: ngày 8 tháng 12 năm 2021

1.2. Tổng quan dự án

Ngày nay, việc ứng dụng công nghệ thông tin và tin học hóa được coi là một trong những yếu tố quyết định đến hoạt động của chính phủ và công ty, đóng vai trò vô cùng quan trọng và có thể tạo ra những bước đột phá mạnh mẽ.

Qua Internet, chúng ta đang làm nhiều việc nhanh hơn so với các phương pháp truyền thống, thúc đẩy sự phát triển của thương mại điện tử, thay đổi đáng kể bộ mặt văn hóa, nâng cao chất lượng cuộc sống con người.

Trong hoạt động kinh doanh, thương mại điện tử hiện nay đã và đang đóng vai trò thúc đẩy, thúc đẩy sự phát triển của doanh nghiệp. Đối với việc quảng bá, giới thiệu sản phẩm mới thì việc đáp ứng nhu cầu của khách hàng là điều cần thiết, vì vậy việc thiết lập một website để thúc đẩy việc bán hàng là vô cùng quan trọng.

Mong muốn tìm hiểu thêm về lĩnh vực kiểm thử phần mềm cũng như trở thành kỹ sư kiểm thử phần mềm sau khi tốt nghiệp đại học, chúng em chọn chủ đề "Kiểm thử trang web bán sách và thiết bị học tập". Trong quá trình làm đồ án, do thời gian và kinh nghiệm thực tế có hạn nên chúng em rất mong nhận được những lời khuyên chân thành từ các thầy cô và các bạn.

1.3. Thực trạng

Như chúng ta đã thấy, từ thời xa xưa khi mà đã xuất hiện sách vở thì đã khai sinh ra khái niệm mua bán sách. Các hàng quán, thư viện bán sách mọc lên rất nhiều nhưng người muốn mua sách thì phải đến tận các tiệm đó để mua sách, mà chưa chắc đã tìm được những loại sách mà mình mong muốn hoặc là đã hết hàng. Hiện nay, công nghệ thông tin càng phát triển và dịch bệnh hoành hành thì nhu cầu của con người ngày càng cao, họ muốn một thứ gì đó thuận tiện và nhanh chóng hơn những phương pháp truyền thống.

1.4. Định nghĩa vấn đề

Các vấn đề của mua bán sách trực tiếp:

- Mua bán sách trực tiếp phải ra trực tiếp
- Phải lựa chọn sách
- Nỗi lo về hết sách cần mua
- Ít chương trình giảm giá

1.5. Giải pháp đề xuất

Các giải pháp đề xuất khi mua bán sách và thiết bị học tập online:

- Mua hàng tại nhà
- Có thể biết được sản phẩm hết hay còn hàng
- Lựa chọn sách, thiết bị học tập nhanh chóng với công cụ tìm kiếm, lọc
- Thanh toán trực tuyến nhanh với thẻ tín dụng
- Theo dõi trạng thái đơn hàng
- Thông kê doanh thu hệ thống
- Thông kê sản phẩm nào còn hàng, sản phẩm nào hết hàng để biết và thêm sản phẩm đó vào kho

1.5.1 Chức năng

Đối với khách hàng:

- Cho phép người dùng tìm kiếm, lọc ra những sản phẩm người dùng có nhu cầu.
- Cho phép người dùng đặt hàng từ xa và thanh toán trả trước với thẻ tín dụng.
- Cho phép người dùng liên hệ người sở hữu hệ thống website.
- Cho phép người dùng đánh giá sản phẩm
- Cho phép người dùng theo dõi đơn hàng, giỏ hàng...
- Và một số chức năng thường có ở người dùng website như: đăng ký, đăng nhập, quên mật khẩu tài khoản, chỉnh sửa thông tin tài khoản, cập nhật mật khẩu...

Đối với quản trị viên:

- Cho phép xem các số liệu thống kê trên hệ thống: thống kê tổng doanh thu, thống kê doanh thu hàng tháng, thống kê số lượt đặt hàng, thống kê tổng số sản phẩm, số lượng sản phẩm còn hàng - hết hàng, thống kê tổng người dùng trên hệ thống.
- Cho phép quản lý sản phẩm: xem danh sách tất cả sản phẩm trên hệ thống, xóa sản phẩm, sửa thông tin sản phẩm, thêm số lượng sản phẩm, tạo sản phẩm mới.
- Cho phép quản lý người dùng: xem danh sách tất cả người dùng, xóa người dùng, chỉnh sửa quyền hạn người dùng.
- Cho phép quản lý đơn hàng: xem danh sách tất cả đơn hàng trên hệ thống, chỉnh sửa trạng thái đơn hàng, xóa đơn hàng.
- Cho phép quản lý các bình luận đánh giá: xem danh sách tất cả bình luận của 1 sản phẩm, xóa bình luận không hợp lệ.

1.5.2 Ưu điểm và nhược điểm

Ưu điểm và nhược điểm của giải pháp đề xuất:

Ưu điểm:

- Cho phép quản trị viên quản lý được dữ liệu của hệ thống (doanh thu, người dùng, sản phẩm, đơn hàng, đánh giá) và có thể thêm, xóa, cập nhật dữ liệu.
- Cho phép người dùng thực hiện được quyền lợi khách hàng cơ bản: xem, tìm kiếm, chọn lọc sản phẩm theo ý muốn, đặt hàng nếu có nhu cầu mua và có thể thanh toán ngay lập tức với thẻ tín dụng

Nhược điểm:

- Vẫn còn lỗi trang giới hàng đối với khách hàng mới (có sẵn 1 sản phẩm trong đó)
- Quản trị viên vẫn chưa thực sự quản lý được sản phẩm (thêm, xóa, cập nhật)
- Khách hàng không đổi được thông tin người dùng và password khi cần thay đổi

1.6. Các yêu cầu về chức năng

Các yêu cầu chức năng của hệ thống của chúng tôi được liệt kê như sau:

- Quản trị viên:

- Có các chức năng thêm, xóa, cập nhật sản phẩm.
- Khi khách hàng đặt hàng. Đơn hàng sẽ được admin xử lý.
- Hiển thị thông tin về sản phẩm
- Xử lý giới hàng

- Khách hàng:

- Có thể tìm kiếm các sản phẩm theo tên.
- Quản lý thông tin giới hàng: Khách hàng có thể thêm sản phẩm, xóa hoặc cập nhật lại danh sách các sản phẩm trong giỏ hàng của mình.

2. Test plan

2.1. Mục tiêu kiểm thử

- Mục tiêu kiểm thử ở đây là xác minh xem chức năng của **Website Bán sách và thiết bị học tập** có hoạt động đúng như mong muốn hay không có thực thi đúng các chức năng mà người lập trình đã thực hiện hay không.

- Việc kiểm thử sẽ thực thi và xác minh lại mua - bán sách và thiết bị học tập có chính xác và thực tế hay không, việc quản lý dữ liệu của tài khoản quản trị có hợp lý hay không từ đó đưa ra các biện pháp khắc phục thông qua mức độ nghiêm trọng của kết quả kiểm thử từ đó đưa ra độ ưu tiên khác phục kèm theo chiến lược khắc phục của các lỗi phát sinh phát hiện qua test case.
- Từ đó đưa ra sản phẩm hoàn thiện nhất.

2.2. Nguyên tắc kiểm tra

- Xây dựng công cụ Kiểm tra sẽ được tập trung vào việc xem xét hiệu quả của dự án, chi phí và độ hiệu quả của sản phẩm .
- Các quy trình kiểm tra sẽ được xác định rõ ràng, nhưng linh hoạt, với khả năng thay đổi khi cần thiết.
- Sẽ có tài liệu chung cung cấp bởi người lập trình để xác định và hỗ trợ người kiểm thử .
- Các hoạt động kiểm tra sẽ được xây dựng dựa trên các giai đoạn trước đó để tránh dư thừa hoặc trùng lặp nỗ lực.
- Kiểm tra sẽ được chia thành các giai đoạn rõ ràng, mỗi giai đoạn đều có mục tiêu và mục tiêu được xác định rõ ràng.
- Sẽ có tiêu chí đầu vào và đầu ra.

2.3. Phương pháp tiếp cận dữ liệu

- Web bán sách sẽ bao gồm phần front-end chứa giao diện và phần back-end chứa các code để gắn vào front-end. Ngoài ra còn phần csdl được lưu trữ trên mongodb và web được deploy lên Heroku.
- Khi đó chúng ta sẽ truy cập vào địa chỉ heroku để truy xuất dữ liệu.

2.4. Phạm vi và mức độ kiểm tra

- **Mục tiêu :** mục đích của thử nghiệm này là để đảm bảo các lỗi nghiêm trọng cần được loại bỏ trước khi các cấp thử nghiệm tiếp theo có thể bắt đầu.
- **Phạm vi:** Kiểm tra lỗi xảy ra ở phần Back-end và Front-end
- **Tester:** Nhóm kiểm thử.

- **Phương pháp:** kiểm thử khám phá này được thực hiện trong ứng dụng mà không có bất kỳ tập lệnh và tài liệu thử nghiệm nào.
- **Thời gian:** Mỗi khi bắt đầu một chu kỳ kiểm thử.

2.4.1. Khám phá

- **Mục đích:** Kiểm thử chức năng sẽ được thực hiện để kiểm tra các chức năng của website. Kiểm tra chức năng được thực hiện bằng cách cung cấp đầu vào và xác nhận đầu ra từ ứng dụng.
- **Phạm vi:** Các chức năng của website.
- **Tester:** Nhóm kiểm thử.
- **Phương thức:** Phần kiểm thử chức năng cần có tài liệu kiểm thử.
- **Thời gian:** Sau khi kiểm thử Khám phá hoàn thành.
- **Tiêu chí kiểm thử:**
 - Tài liệu Đặc điểm kỹ thuật chức năng đã được phê duyệt, tài liệu Use case phải có sẵn trước khi bắt đầu giai đoạn thiết kế Thủ nghiệm.
 - Các trường hợp thử nghiệm đã được phê duyệt và chấp thuận trước khi bắt đầu thực hiện Thủ nghiệm.
 - Đã hoàn thành phát triển, đơn vị được kiểm tra với trạng thái đạt và kết quả được chia sẻ cho nhóm Kiểm tra để tránh các khuyết tật trùng lặp.
 - Môi trường thử nghiệm với ứng dụng được cài đặt, cấu hình và trạng thái sẵn sàng sử dụng.

2.4.2. Kiểm thử chức năng

Bảng Giao Công Việc Test

S.No.	Công việc được giao	Người thực hiện	Reviewer
1	Kế hoạch kiểm thử	Test Lead	
2	Các trường hợp kiểm thử chức	Test Team	

	năng		
3	Lỗi ghi nhật ký trong HP ALM	Test Team	
4	Báo cáo trạng thái hàng ngày / hàng tuần	Test Team/ Test Lead	
5	Báo cáo kiểm tra kết thúc	Test Lead	

Table 2: Bảng Giao Công Việc Test

2.4.3. Kiểm tra sự chấp nhận của người dùng (UAT)

- **Mục đích:** kiểm tra này tập trung vào việc xác nhận logic nghiệp vụ. Nó cho phép người dùng cuối hoàn thành một đánh giá cuối cùng của hệ thống trước khi triển khai.
- **Người kiểm thử:** UAT được thực hiện bởi người dùng cuối
- **Phương pháp:** Cho phép người dùng sử dụng và từ đó lấy các thông tin phản hồi từ khách hàng.
- **Thời gian:** Sau khi tất cả các cấp độ kiểm tra khác (Khám phá và Chức năng) được thực hiện. Chỉ sau khi hoàn thành thử nghiệm này, sản phẩm mới có thể được đưa vào sử dụng.

3. Chiến lược thực hiện

3.1. Tiêu chí đầu vào và đầu ra

Tiêu chí thoát	Nhóm kiểm tra	Đội kỹ thuật	Ghi chú
100% tập lệnh thử nghiệm được thực thi			
Tỷ lệ vượt qua 95% của Tập lệnh thử nghiệm			
Không có khiếm khuyết nghiêm trọng và mức độ nghiêm trọng cao mở			

95% các lỗi ở mức độ trung bình đã được đóng lại			
Tất cả các khiếm khuyết còn lại đều bị hủy bỏ hoặc được ghi lại dưới dạng Yêu cầu thay đổi cho bản phát hành trong tương lai			
Tất cả các khiếm khuyết còn lại đều bị hủy bỏ hoặc được ghi lại dưới dạng Yêu cầu thay đổi cho bản phát hành trong tương lai			
Tất cả các kết quả mong đợi và thực tế đều được ghi lại và ghi lại bằng tập lệnh thử nghiệm			
Tất cả các chỉ số kiểm tra được thu thập dựa trên báo cáo từ HP ALM			
Đã hoàn thành bản ghi nhớ kiểm tra và ký kết			
Đã hoàn thành dọn dẹp môi trường thử nghiệm và sao lưu môi trường mới			

Table 3: Tiêu chí kiểm thử

3.2. Chu kỳ kiểm tra

- Sẽ có hai chu kỳ để kiểm tra chức năng. Mỗi chu kỳ sẽ thực thi tất cả các tập lệnh.
- Mục tiêu của chu trình đầu tiên là xác định bất kỳ lỗi nào bị tắc nghẽn, khuyết điểm nghiêm trọng và hầu hết các khuyết điểm cao. Nó được mong đợi sử dụng một số công việc xung quanh để có được tất cả các tập lệnh.
- Mục tiêu của chu kỳ thứ hai là xác định các khuyết tật ở mức cao và trung bình còn lại, loại bỏ các công việc xung quanh từ chu kỳ đầu tiên, sửa các khoảng trống trong kịch bản và thu được kết quả hoạt động.

3.3. Xác thực và quản lý sai sót

Mức độ nghiêm trọng	Sự va chạm
1 (Nghiêm trọng)	<ul style="list-style-type: none"> § Lỗi này đủ nghiêm trọng để làm hỏng hệ thống, gây hỏng tệp hoặc gây mất dữ liệu tiềm ẩn. § Nó gây ra sự trở lại bất thường của hệ điều hành (sự cố hoặc thông báo lỗi hệ thống xuất hiện). § Nó khiến ứng dụng bị treo và yêu cầu khởi động lại hệ thống.
2 (Cao)	<ul style="list-style-type: none"> § Nó gây ra thiếu chức năng quan trọng của chương trình với cách giải quyết.
3 (Trung bình)	<ul style="list-style-type: none"> § Lỗi này sẽ làm giảm chất lượng của Hệ thống. Tuy nhiên, có một giải pháp thông minh để đạt được chức năng mong muốn - ví dụ: thông qua một màn hình khác. § Lỗi này ngăn các khu vực khác của sản phẩm được kiểm tra. Tuy nhiên các khu vực khác có thể được kiểm tra độc lập.
4 (Thấp)	<ul style="list-style-type: none"> § Có thông báo lỗi không đầy đủ hoặc không rõ ràng, ảnh hưởng tối thiểu đến việc sử dụng sản phẩm.
5(Cosmetic)	<ul style="list-style-type: none"> § Có thông báo lỗi không đầy đủ hoặc không rõ ràng, không ảnh hưởng đến việc sử dụng sản phẩm.

Table 4: Mức độ nghiêm trọng của lỗi

3.4. Số liệu kiểm tra

- Các thước đo kiểm tra để đo lường tiến độ và mức độ thành công của kiểm thử sẽ được phát triển và chia sẻ với người quản lý dự án để phê duyệt. Dưới đây là một số chỉ số.

Báo cáo	Miêu tả	Tần suất
Chuẩn bị kiểm tra & Trạng thái Thực thi	-Báo cáo về% hoàn thành,% WIP,% đạt,% không thành công .	-Hằng tuần / Hằng ngày.

	<ul style="list-style-type: none"> -Khuyết điểm mức độ nghiêm trọng Trạng thái - Mở, đóng, bất kỳ hình thức nào khác . -Trạng thái. 	
Trạng thái thực hiện hàng ngày	<ul style="list-style-type: none"> - Báo cáo về Đạt, Không đạt, Tổng số khiếm khuyết, Đánh dấu Showstopper . - Lỗi nghiêm trọng. 	-Hàng ngày.
Báo cáo tình trạng hàng tuần của dự án	<ul style="list-style-type: none"> -Báo cáo theo định hướng dự án (Theo yêu cầu của PM) 	-Nhóm dự án hàng tuần nếu cần cập nhật hàng tuần ngoài hàng ngày và có sẵn mẫu với nhóm dự án để sử dụng.

Table 5: *Số liệu kiểm tra sản phẩm*

CHƯƠNG II: CÁC PHƯƠNG PHÁP KIỂM THỬ

1. Kiểm thử đơn vị – Unit test

Là một loại kiểm thử phần mềm trong đó thực hiện kiểm thử từng đơn vị hoặc thành phần riêng lẻ của phần mềm. Mục đích của việc Kiểm thử đơn vị là để xác nhận rằng của mỗi đơn vị hay mã code của phần mềm thực hiện chức năng của chúng đúng như mong đợi. Kiểm thử đơn vị được thực hiện trong quá trình phát triển (giai đoạn thực hiện code) của một ứng dụng và được thực hiện bởi các kỹ sư phần mềm. Kiểm thử đơn vị sẽ thực hiện kiểm thử độc lập một phần code và xác minh tính chính xác của nó. Một đơn vị có thể là một chức năng, một phương thức, thủ tục, mô-đun hoặc đối tượng riêng lẻ.

Trong SDLC (Software Development Life-cycle - Vòng đời phát triển phần mềm), STLC (Software Testing Life Cycle - Quy trình kiểm thử phần mềm), V Model (Mô hình chữ V), kiểm thử đơn vị là cấp độ kiểm thử đầu tiên được thực hiện trước khi kiểm thử tích hợp. Kiểm thử đơn vị là một kỹ thuật kiểm tra WhiteBox thường được thực hiện bởi kỹ sư phần mềm. Mặc dù, trên thực tế do giới hạn về thời gian mà những kỹ sư phần mềm khó có thể thực hiện kiểm thử đơn vị, lúc này việc Kiểm thử đơn vị được thực hiện bởi những QA (Quality Assurance/Tester).

Đôi khi các kỹ sư phần mềm cố gắng tiết kiệm thời gian bằng cách thực hiện kiểm tra đơn vị một cách tối thiểu nhất. Việc bỏ qua kiểm tra đơn vị dẫn đến chi phí sửa lỗi cao hơn (nếu xảy ra lỗi) trong quá trình Kiểm thử hệ thống, Kiểm thử tích hợp và Kiểm thử Beta khi ứng dụng được hoàn thành. Việc kiểm thử đơn vị một cách thích hợp được thực hiện trong giai đoạn phát triển phần mềm giúp tiết kiệm cả thời gian và tiền bạc. Đây là những lý do chính để thực hiện kiểm tra đơn vị.

Lợi ích của việc kiểm thử đơn vị:

- Kiểm thử đơn vị giúp sửa lỗi sớm trong chu kỳ phát triển phần mềm, từ đó giúp tiết kiệm thời gian và chi phí.
- Giúp các kỹ sư phần mềm hiểu cơ sở code và cho phép họ thực hiện các thay đổi nhanh chóng
- Kiểm thử đơn vị có thể được dùng như tài liệu dự án
- Kiểm thử đơn vị hỗ trợ việc sử dụng lại code. Khi có dự án mới, các mã code và cách Kiểm thử đơn vị của dự án cũ có thể được tái sử dụng, chỉ cần điều chỉnh các mã code và cách test sao cho phù hợp.

2. Kiểm thử tích hợp – Intergration Test

KIỂM THỬ TÍCH HỢP được định nghĩa là một loại kiểm thử trong đó các mô-đun (modules) phần mềm được tích hợp một cách hợp lý và được thử nghiệm dưới dạng một nhóm. Một dự án phần mềm điển hình bao gồm nhiều mô-đun phần mềm, được mã hóa bởi các lập trình viên khác nhau. Mục đích của cấp độ kiểm tra này là để lộ ra các khiếm khuyết (lỗi) trong tương tác giữa các mô-đun phần mềm khi chúng được tích hợp với nhau.

Kiểm thử tích hợp tập trung vào kiểm tra giao tiếp dữ liệu giữa các mô-đun. Do đó, nó cũng được gọi là 'I & T' (Tích hợp và Kiểm tra), 'Kiểm tra chuỗi' và đôi khi là 'Kiểm tra luồng'.

Mặc dù từng mô-đun phần mềm sẽ được Unit Test (kiểm thử đơn vị), nhưng lỗi vẫn sẽ tồn tại vì nhiều lý do như:

- Một mô-đun, nhìn chung, được thiết kế bởi một nhà phát triển phần mềm độc lập mà có sự hiểu biết và lập trình logic có thể khác với các lập trình viên khác. Kiểm thử tích hợp là cần thiết để xác minh các mô-đun phần mềm hoạt động cùng nhau một cách nhất quán.
- Tại thời điểm phát triển mô-đun, sẽ có nhiều lúc khách hàng đưa ra những thay đổi về yêu cầu. Các yêu cầu mới này có thể sẽ không được Unit Test (kiểm thử đơn vị) và do đó Kiểm thử tích hợp hệ thống trở nên cần thiết.
- Các giao diện của các mô-đun phần mềm cùng với cơ sở dữ liệu có thể sẽ bị lỗi
- Giao diện phần cứng bên ngoài, nếu có, có thể bị lỗi
- Xử lý những trường hợp ngoại lệ một cách không phù hợp có thể gây ra các vấn đề khác

3. Kiểm thử hệ thống – System Test

Kiểm thử hệ thống hay còn gọi là System Testing, là kiểm tra lại toàn bộ hệ thống sau khi tích hợp. Nó cho phép kiểm tra sự tuân thủ của hệ thống theo yêu cầu. Loại kiểm thử này kiểm tra sự tương tác tổng thể của các thành phần. Nó liên quan đến tải, hiệu suất, độ tin cậy và kiểm tra bảo mật.

System Testing sẽ được thực hiện sau integration testing. Đây là một bước giữ vai trò quan trọng trong việc cho ra đời một sản phẩm chất lượng cao.

System Testing thuộc loại kiểm thử hộp đen(Black Box Testing), là một phương pháp kiểm thử phần mềm dựa trên đầu vào và đầu ra của chương trình để test mà không cần quan tâm code bên trong của phần mềm ra sao. Mục đích chính là liệu nó có đáp ứng được sự mong đợi của người dùng hay không?

- Kiểm thử phần mềm là khâu vô cùng quan trọng trong quá trình phát triển 1 sản phẩm công nghệ. Nó chỉ ra lỗi và sai sót đã được thực hiện trong các giai đoạn phát triển.
- System testing đảm bảo độ tin cậy của khách hàng và sự hài lòng của họ về ứng dụng mà mình tạo ra
- Giúp tăng hiệu suất công việc do giảm được tối đa thời gian để tìm lỗi trên ứng dụng phần mềm hoặc sản phẩm nhiều lần
- Kiểm thử phần mềm là cần thiết vì nó giúp cung cấp các ứng dụng phần mềm cho khách hàng phân phối được hướng sản phẩm chất lượng cao hoặc chi phí bảo trì ứng dụng phần mềm thấp hơn, tiết kiệm hơn và do đó dẫn đến hiệu quả cao nhất và đáng tin cậy hơn.
- Được thực hiện trong một môi trường tương tự như môi trường production, cho phép các nhà phát triển cũng như các bên liên quan có được ý tưởng về phản ứng của người dùng đối với sản phẩm.
- Quá trình này đặc biệt đảm bảo rằng ứng dụng không dẫn đến bất kỳ lỗi nào, hạn chế tối đa những tồn kém trong tương lai hoặc trong các giai đoạn của quá trình phát triển sản phẩm.

4. Kiểm thử chấp nhận sản phẩm – Acceptance Test

Acceptance Testing (Kiểm thử chấp nhận) là một kiểm thử nhằm xác định hệ thống phần mềm có đạt yêu cầu kỹ thuật hay không. Bằng việc kiểm tra các hành vi của hệ thống qua dữ liệu thực tế, kiểm thử chấp nhận sẽ xác định có hay không việc hệ thống đáp ứng được các tiêu chí lẩn yêu cầu của khách hàng. Một số kỹ thuật được sử dụng trong Acceptance Testing đó là phân tích giá trị biên giới, phân vùng tương đương và sử dụng bảng quyết định.

Nhờ Acceptance Testing mà bạn có thể xác định được giải pháp, phần mềm tạo ra đã đi đúng hướng mà khách hàng đề xuất hay không. Ngoài ra, kiểm thử chấp nhận còn mang lại rất nhiều lợi ích khác như:

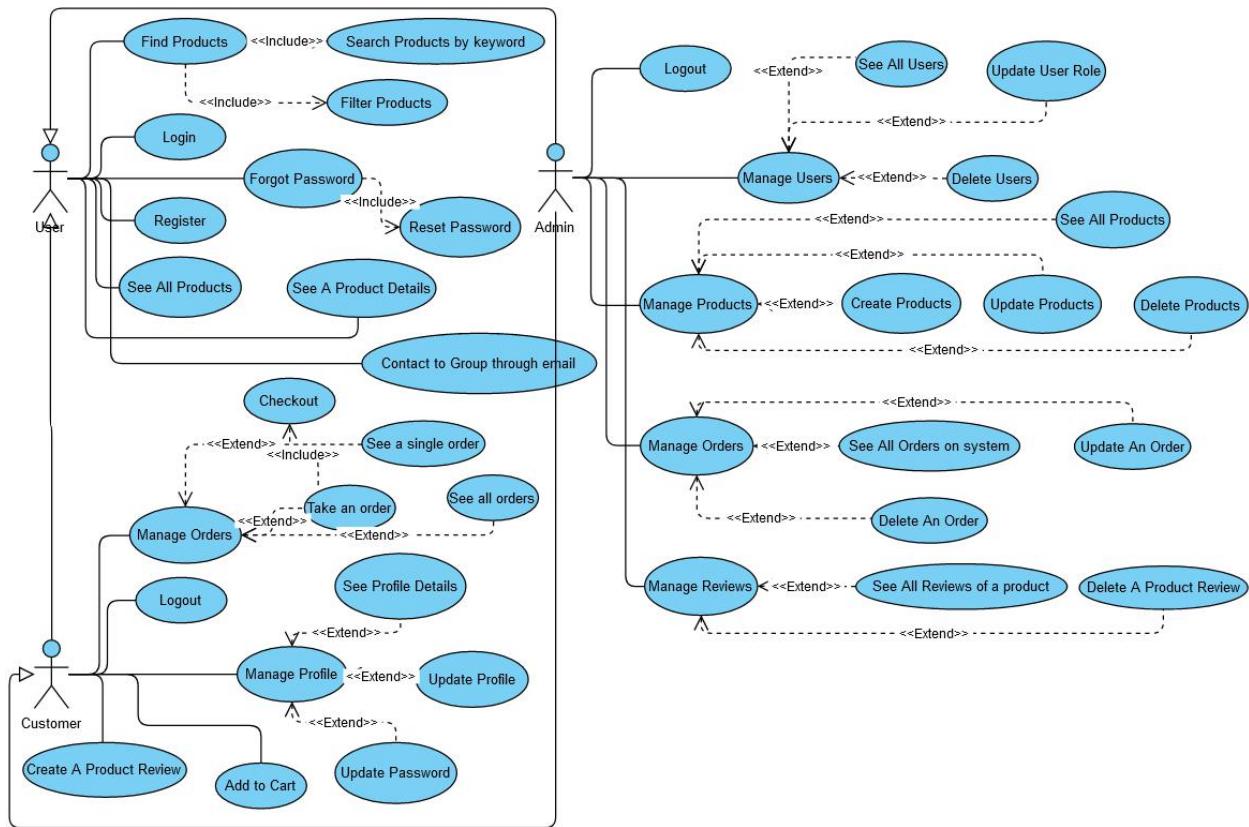
- Acceptance Testing giúp tìm hiểu và xác định các yêu cầu của người dùng bằng cách kiểm chứng trực tiếp.
- Thông qua Acceptance Testing sẽ tìm được những vấn đề ở Unit hoặc Integration Test đã để lọt.
- Acceptance Testing giúp bạn có cái nhìn tổng quan nhất về kết quả hệ thống đạt được.
- Acceptance Testing được sử dụng để xác định và xác minh nhu cầu của khách hàng.

Kiểm thử chấp nhận gồm các loại như sau:

- Alpha & Beta Testing
- Contract Acceptance Testing
- Regulation Acceptance Testing
- Operational acceptance Testing
- Black Box Testing

CHƯƠNG III: MÔ HÌNH HOÁ YÊU CẦU

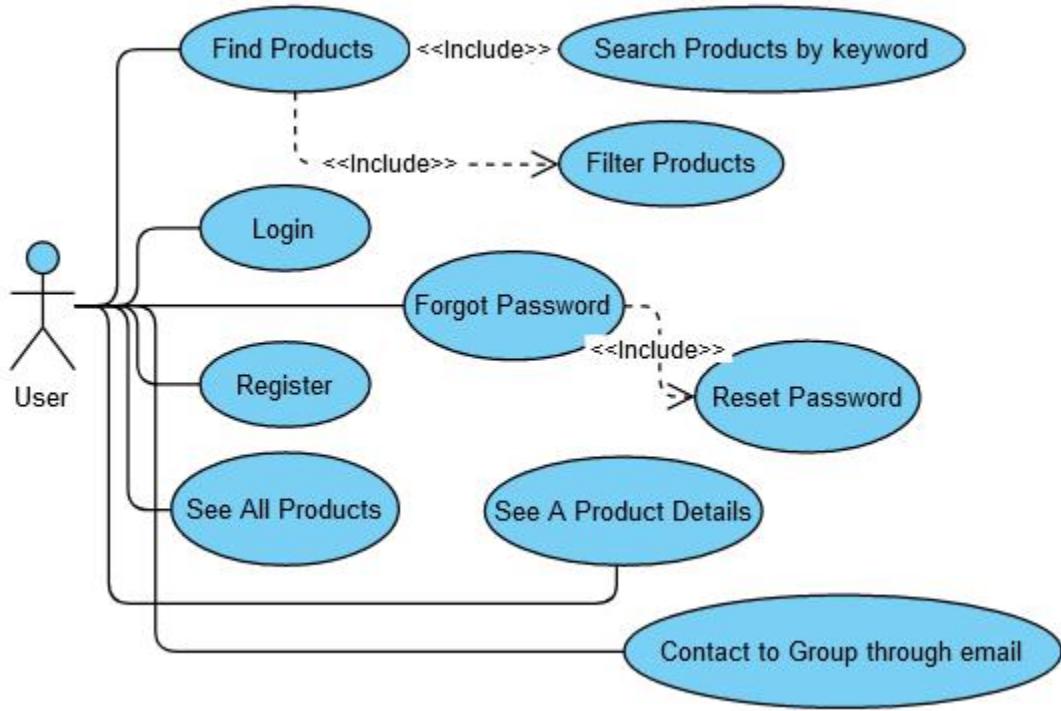
1. Usecase diagram



Hình 1: Sơ đồ usecase tổng

Figure 3.1: Usecase

2. Chi tiết use case



Hình 2: Usecases của Actor User

Figure 3.2.1: User

2.1. Register

USE CASE – UC_01			
Use Case No.	UC_01	Use Case Version	1.0
Use Case Name	Register		
Author	Bùi Hà Nhi		
Date	23/11/2021	Priority	High

Actor: User

Summary: Cho phép người dùng đăng ký tài khoản.

Goal: Người dùng có thể tạo tài khoản với vai trò khách hàng.

Triggers: Người dùng gửi lệnh đăng ký bằng cách nhấn vào icon user tại danh mục.

Preconditions: Người dùng phải có địa chỉ email mới, password tài khoản phải hơn 9 ký tự.

Post conditions: Thành công: Đã tạo tài khoản mới.

Thất bại: Hệ thống hiển thị thông báo lỗi.

Step:

Step	Actor Action	System Response
1	Người dùng nhập username	
2	Người dùng nhập email	
3	Người dùng nhập password	
4	Người dùng nhấn vào button “Browse...”	
5	Người dùng nhấn vào 1 ảnh	
6	Người dùng nhấn vào button “Open”	
7	Người dùng nhấn vào button “Register”	Hệ thống tạo một tài khoản mới với thông tin đầu vào và vai trò người dùng là khách hàng. Hệ thống chuyển hướng đến trang đăng nhập.

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
2	Người dùng không nhập thông tin và không chọn avatar mà nhấn button “Register”	Hệ thống thông báo “Please fill out this field” tại khung nhập thông tin trống đó	
3	Người dùng nhập email không đúng định dạng, ví dụ ‘buihanhi002’	Hệ thống thông báo “Please include an ‘@’ in the email address. ‘Buihanhi002’ is missing an ‘@’”	
4	Người dùng nhập thông tin và không chọn avatar mà nhấn button “Register”	Hệ thống thông báo “Internal Server Error”	
5	Người dùng nhập email đã dùng để đăng ký trước đó	Hệ thống hiển thị thông báo “User validation failed: Email: Email is available. Please register with another email”	
6	Người dùng nhập password không đủ 8 ký tự	Hệ thống hiển thị thông báo “User validation failed: Password: Password should have more than 8 characters”	
7	Người dùng chọn avatar không đúng định dạng file ảnh	Hệ thống hiển thị thông báo “User validation failed: Unsupported Zip File”	

Business Rules:

- Mật khẩu: độ dài của mật khẩu tối thiểu 8 ký tự.
- Hệ thống phải đảm bảo rằng không ai có thể đọc được mật khẩu của người dùng.
- Mật khẩu phải được mã hóa trong cơ sở dữ liệu.
- Trường email phải đúng định dạng.
- Avatar phải đúng định dạng Image File.

*Table 6: Đặc tả usecase đăng ký***2.2. Login**

USE CASE – UC_02			
Use Case No.	UC_02	Use Case Version	1.0
Use Case Name	Login		
Author	Bùi Hà Nhi		
Date	23/11/2021	Priority	High

Actor: User

Summary: Cho phép người dùng đăng nhập tài khoản.

Goal: Người dùng có thể đăng nhập vào hệ thống với vai trò cụ thể.

Triggers: Người dùng gửi lệnh đăng nhập bằng cách nhấp vào icon user tại danh mục.

Preconditions: Người dùng phải có tài khoản đã đăng ký thành công trước đó.

Post conditions: Thành công: Người dùng được ủy quyền chính xác với vai trò cụ thể trong hệ thống.

Thất bại: Hệ thống hiển thị thông báo lỗi.

Step:

Step	Actor Action	System Response
1	Người dùng nhập email	
2	Người dùng nhập password	
3	Người dùng nhấp vào button “Login”	Người dùng được ủy quyền với vai trò của mình. Hệ thống chuyển hướng đến trang thông tin cá nhân của tài khoản vừa đăng nhập.

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

2	Người dùng nhập sai thông tin xác thực bằng email và password	Hệ thống thông báo “Invalid email or password”	
3	Người dùng không nhập email và password mà nhấn button “Login”	Hệ thống thông báo “Please fill out this field” tại khung nhập thông tin trống đó	
4	Người dùng nhập email không đúng định dạng, ví dụ ‘buihanhi002’	Hệ thống thông báo “Please include an ‘@’ in the email address. ‘Buihanhi002’ is missing an ‘@’”	

Business Rules:

- Mật khẩu: độ dài của mật khẩu tối thiểu 8 ký tự và khớp với lúc đăng ký tài khoản
- Mật khẩu phải được mã hóa trong cơ sở dữ liệu.
- Trường email phải đúng định dạng và khớp với lúc đăng ký tài khoản

Table 7: *Đặc tả usecase đăng nhập*

2.3. Forgot Password

USE CASE – UC_03			
Use Case No.	UC_03	Use Case Version	1.0
Use Case Name	Forgot Password		
Author	Bùi Hà Nhi		
Date	23/11/2021	Priority	High

Actor: User

Summary: Cho phép người dùng gửi email tài khoản mình quên mật khẩu đến hệ thống và nhận lại mật khẩu tạm thời mới.

Goal: Người dùng nhận được mật khẩu mới.

Triggers: Người dùng gửi lệnh quên mật khẩu bằng cách nhấn vào “Forgot Password?” tại trang Đăng nhập.

Preconditions: Người dùng phải có địa chỉ email được xác thực trên dịch vụ gmail của google.

Post conditions: Thành công: Người dùng nhận được mật khẩu mới.

Thất bại: Người dùng không nhận được mật khẩu mới.

Step:

Step	Actor Action	System Response
1	Người dùng nhập email	
2	Người dùng nhấn vào button “Send”	Hệ thống sẽ gửi mail đến gmail người dùng vừa nhập một password tạm thời mới được mã hóa. Hệ thống chuyển hướng đến trang Cài lại mật khẩu.

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

2	Người dùng không gmail mà nhấn button “Send”	Hệ thống thông báo “Please fill out this field” tại khung nhập thông tin trống đó	
3	Người dùng nhập email không đúng định dạng, ví dụ ‘buihanhi002’	Hệ thống thông báo “Please include an ‘@’ in the email address. ‘Buihanhi002’ is missing an ‘@’”	
4	Người dùng nhập email không phải là của mình, nhưng đã được đăng ký thành công trên hệ thống	Hệ thống vẫn gửi mail đến email đó nhưng người dùng sẽ không sử dụng được mật khẩu hệ thống gửi.	
5	Người dùng nhập email chưa được xác thực với dịch vụ google	Hệ thống hiển thị thông báo “User not found”	

Business Rules:

- Hệ thống phải đảm bảo rằng không ai có thể đọc được mật khẩu của người dùng.
- Mật khẩu phải được mã hóa.
- Trường email phải đúng định dạng.
- Mail gửi mật khẩu mới tạm thời phải gửi đúng cho email được nhập.

Table 8: Đặc tả usecase quên mật khẩu

2.4. Reset Password

USE CASE – UC_04			
Use Case No.	UC_04	Use Case Version	1.0
Use Case Name	Reset Password		
Author	Bùi Hà Nhi		

Date	23/11/2021	Priority	High
------	------------	----------	------

Actor: User

Summary: Cho phép người dùng cài lại mật khẩu mới cho tài khoản.

Goal: Người dùng có thể sử dụng tài khoản của mình với mật khẩu mới.

Triggers: Người dùng gửi lệnh cài lại mật khẩu bằng cách nhấn vào button “Send” tại trang Quên mật khẩu.

Preconditions: Người dùng đã nhập email tại trang Đăng nhập trước khi nhấn “Forgot Password?”, người dùng phải có password tạm thời được gửi mail từ hệ thống.

Post conditions: Thành công: Đã cài lại mật khẩu mới.

Thất bại: Hệ thống hiển thị thông báo lỗi.

Step:

Step	Actor Action	System Response
1	Người dùng nhập password mới	
2	Người dùng nhập password xác nhận lại	
3	Người dùng nhấn vào button “Reset Password”	Hệ thống xác nhận tài khoản với mật khẩu mới. Hệ thống tự động đăng nhập vào. Hệ thống chuyển hướng đến trang thông tin cá nhân của tài khoản.

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	

2	Người dùng không nhập thông tin mà nhấn button “Reset Password”	Hệ thống thông báo “Please fill out this field” tại khung nhập thông tin trống đó	
3	Người dùng nhập password không đủ 8 ký tự	Hệ thống hiển thị thông báo “User validation failed: Password: Password should have more than 8 characters”	
4	Người dùng nhập password xác nhận không khớp với password	Hệ thống hiển thị thông báo “User validation failed: Password: Confirm Password is different from Password”	

Business Rules:

- Mật khẩu: độ dài của mật khẩu tối thiểu 8 ký tự.
- Hệ thống phải đảm bảo rằng không ai có thể đọc được mật khẩu của người dùng.
- Mật khẩu phải được mã hóa trong cơ sở dữ liệu.

Table 9: Đặc tả usecase cài lại mật khẩu

2.5. See All Products

USE CASE – UC_05			
Use Case No.	UC_05	Use Case Version	1.0
Use Case Name	See All Products		
Author	Bùi Hà Nhi		
Date	23/11/2021	Priority	High

Actor: User

Summary: Cho phép người dùng xem được tất cả các sản phẩm trên hệ thống.

Goal: Người dùng có thể xem danh sách các sản phẩm trên hệ thống.

Triggers: Người dùng gửi lệnh xem tất cả các sản phẩm bằng cách nhấn vào “Products” tại danh mục.

Preconditions: None.

Post conditions: Thành công: Màn hình hiển thị danh sách các sản phẩm trên hệ thống.

Thất bại: Màn hình hiển thị trang loading.

Step:

Step	Actor Action	System Response
1	Người dùng cuộn chuột xuống	
2	Người dùng chọn button số x. Ví dụ: trang số 2	Hệ thống chuyển hướng đến trang các sản phẩm ở trang số 2
3	Người dùng chọn button “Next”	Hệ thống chuyển hướng đến trang các sản phẩm ở trang số kế tiếp trang hiện tại 1 đơn vị
4	Người dùng chọn button “Last”	Hệ thống chuyển hướng đến trang các sản phẩm ở trang cuối cùng
5	Người dùng chọn button “Prev”	Hệ thống chuyển hướng đến trang các sản phẩm ở trang số trước trang hiện tại 1 đơn vị
6	Người dùng chọn button “1st”	Hệ thống chuyển hướng đến trang các sản phẩm ở trang đầu tiên

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	

1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
Business Rules:			
<ul style="list-style-type: none"> • None. 			

Table 9: Đặc tả usecase Xem tất cả các sản phẩm

2.6. See A Product Details

USE CASE – UC_06			
Use Case No.	UC_06	Use Case Version	1.0
Use Case Name	See A Product Details		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: User

Summary: Cho phép người dùng xem thông tin chi tiết của 1 sản phẩm.

Goal: Người dùng có thể xem thông tin chi tiết của 1 sản phẩm.

Triggers: Người dùng gửi lệnh xem thông tin chi tiết của 1 sản phẩm bằng cách nhấp vào 1 sản phẩm trong danh sách các sản phẩm trên hệ thống.

Preconditions: Sản phẩm phải có đầy đủ các thông tin.

Post conditions: Thành công: Xem được thông tin chi tiết của các sản phẩm.

Thất bại: Hệ thống hiển thị trang loading.

Step:

Step	Actor Action	System Response
1	Người dùng xem thông tin chi tiết sản phẩm	

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

2	Người dùng chọn button “Submit Review”	Hệ thống thông báo “Please login to be able to access the resource ”	
3	Người dùng chọn button “Add to Cart”	Hệ thống thông báo “Please login to be able to access the resource ”	

Business Rules:

- None

Table 10: Đặc tả usecase Xem chi tiết thông tin 1 sản phẩm

2.7. Contact to Group through email

USE CASE – UC_07			
Use Case No.	UC_07	Use Case Version	1.0
Use Case Name	Contact to Group through email		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: User

Summary: Cho phép người dùng liên hệ với nhóm đề tài.

Goal: Người dùng có thể liên hệ với nhóm đề tài.

Triggers: Người dùng gửi lệnh liên hệ với nhóm đề tài bằng cách nhấp vào “Contact” tại danh mục.

Preconditions: None.

Post conditions: Thành công: Hiển thị cửa sổ Gmail trong máy tính.

Thất bại: Hệ thống hiển thị thông báo lỗi.

Step:

Step	Actor Action	System Response
1	Người dùng nhấp vào Contact trong danh mục	
2	Người dùng nhấp vào 1 trong các email thành viên để liên hệ nhóm đề tài	Hệ thống chuyển tiếp sang cửa sổ Gmail cho người dùng liên hệ đến gmail của nhóm đề tài.

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None

Table 11: Đặc tả usecase Liên hệ nhóm để tài thông qua email

2.8. Search Products by Keyword

USE CASE – UC_08			
Use Case No.	UC_08	Use Case Version	1.0
Use Case Name	Search Products by Keyword		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: User

Summary: Cho phép người tìm kiếm sản phẩm bằng từ khóa.

Goal: Người dùng có thể tìm kiếm sản phẩm bằng từ khóa.

Triggers: Người dùng gửi lệnh tìm kiếm sản phẩm bằng từ khóa bằng cách nhấp vào icon user tại danh mục.

Preconditions: None.

Post conditions: Thành công: Hiển thị sản phẩm có liên quan đến từ khóa.

Thất bại: Không hiển thị sản phẩm có liên quan đến từ khóa.

Step:

Step	Actor Action	System Response
1	Người dùng nhập từ khóa	
2	Người dùng nhấn button “Search”	Hệ thống hiển thị trang chứa danh sách các sản phẩm liên quan đến từ khóa vừa nhập

Alternative Scenario:

Step	Actor Action	System Response
1	Người dùng nhấn button “Search”	Hệ thống hiển thị trang chứa danh sách tất cả các sản phẩm trong hệ thống

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None

Table 12: Đặc tả usecase Tìm kiếm sản phẩm theo từ khóa

2.9. Filter Products

USE CASE – UC_09			
Use Case No.	UC_09	Use Case Version	1.0
Use Case Name	Filter Products		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: User

Summary: Cho phép người dùng lọc sản phẩm trên hệ thống.

Goal: Người dùng có thể lọc sản phẩm trên hệ thống.

Triggers: Người dùng gửi lệnh lọc sản phẩm bằng cách nhấn vào “Products” tại danh mục.

Preconditions: None.

Post conditions: Thành công: Hiển thị sản phẩm theo lọc sản phẩm.

Thất bại: Hệ thống hiển thị trang loading.

Step:

Step	Actor Action	System Response
1	Người dùng chọn lọc sản phẩm theo giá tiền	Hệ thống hiển thị danh sách các sản phẩm theo giá tiền vừa lọc

Alternative Scenario:

Step	Actor Action	System Response
1	Người dùng chọn lọc sản phẩm theo loại sản phẩm	Hệ thống hiển thị danh sách các sản phẩm theo loại sản phẩm vừa chọn

Alternative Scenario:

Step	Actor Action	System Response
1	Người dùng chọn lọc sản phẩm theo đánh giá	Hệ thống hiển thị danh sách các sản phẩm theo đánh giá vừa lọc

Exceptions:

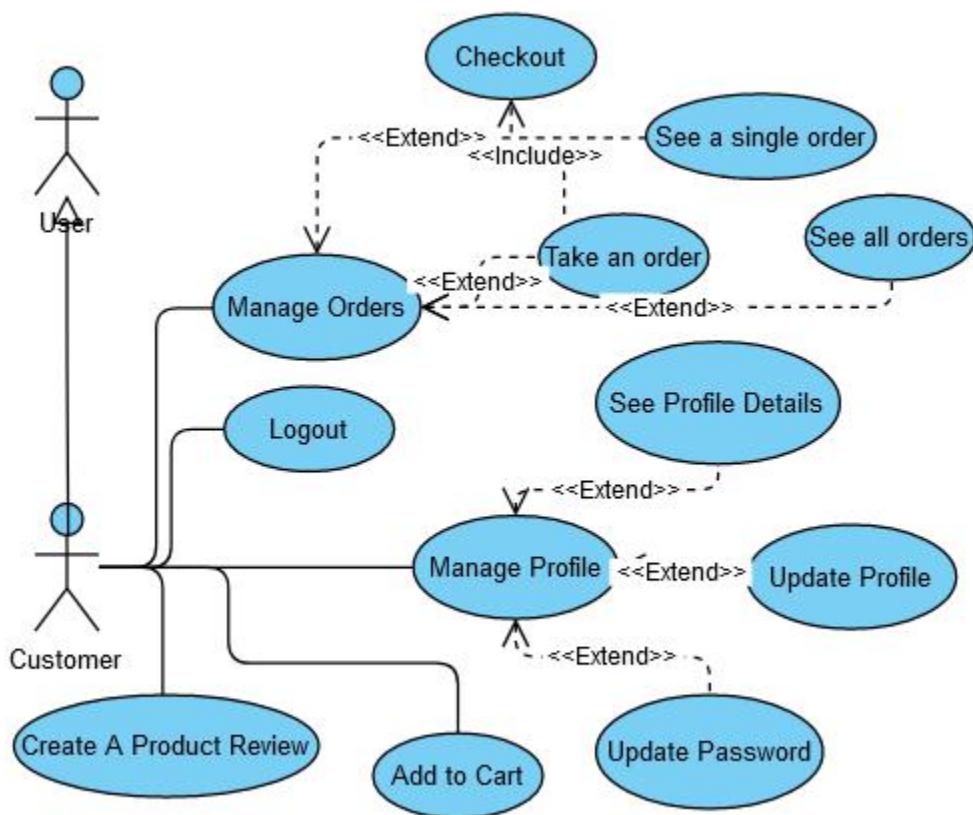
N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 13: Đặc tả usecase Lọc sản phẩm



Hình 3: Usecases của Actor Customer
Figure 3.2.2: Customer

2.10. Log out

USE CASE – UC_10			
Use Case No.	UC_10	Use Case Version	1.0

Use Case Name	Logout
Author	Nguyen Quang Vu

Date	23/11/2021	Priority	High
-------------	------------	-----------------	------

Actor: Customer

Summary: Cho phép người dùng đăng xuất khỏi hệ thống.

Goal: Người dùng đăng xuất thành công.

Triggers: Người dùng nhấp vào nút đăng xuất trên thanh menu của trang chủ.

Preconditions: Khách đã đăng nhập vào hệ thống.

Post conditions: Thành công: Hệ thống chuyển hướng về trang chủ.

Fail: Hệ thống hiển thị thông báo lỗi.

Step:

Step	Actor Action	System Response
1	Người dùng nhấp vào nút "Đăng xuất" trên thanh menu.	Hệ thống xác nhận đăng xuất thành công và quay lại trang chủ.

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response
1		Hệ thống hiển thị thông báo "Vui lòng thử lại"

Table 14: Đặc tả usecase Đăng xuất

2.11. See Profile Details

USE CASE – UC_11			
Use Case No.	UC_11	Use Case Version	1.0
Use Case Name	See Profile Detail		
Author	Nguyen Quang Vu		
Date	23/11/2021	Priority	High
Actor: The trial customer			
Summary: Cho phép người dùng truy cập vào trang cá nhân			
Goal: Người dùng xem trang cá nhân thành công			
Triggers: Người dùng nhấp vào icon profile trên thanh menu của trang chủ.			
Preconditions: N / A			
Post conditions: Thành công: Hệ thống hiển thị kết quả trang cá nhân Thất bại: Hệ thống hiển thị thông báo lỗi			
Step:			
Step	Actor Action	System Response	
1	Nhân viên dùng click icon profile	Hiển thị trang cá nhân	
2	Nhân viên dùng click “edit profile”	Hệ thống chuyển hướng đến trang update profile	

3	Người dùng nhấn “My Orders”	Hệ thống chuyển hướng đến trang order của người dùng
4	Người dùng nhấn “change password”	Hệ thống chuyển hướng đến trang update profile để đổi mật khẩu

Alternative Scenario: N/A.

Exceptions:

No	Actor Action	System Response
1		Hệ thống hiển thị thông báo "Vui lòng kiểm tra kết nối internet của bạn"

Business Rules:

None

Table 15: Đặc tả usecase Xem thông tin cá nhân

2.12. Update Profile

USE CASE – UC_12			
Use Case No.	UC_12	Use Case Version	1.0
Use Case Name	Update Profile		
Author	Nguyen Quang Vu		

Date	23/11/2021	Priority	High
------	------------	----------	------

Actor: Customer

Summary: Cho phép người dùng cập nhật thông tin cá nhân.

Goal: Người dùng cập nhật trang cá nhân thành công.

Triggers: Người dùng nhấp vào icon update trên update profile.

Preconditions: N / A

Post conditions: Thành công: Hệ thống hiển thị kết quả trang thông tin cá nhân

Thất bại: Hệ thống hiển thị thông báo lỗi

Step:

Step	Actor Action	System Response
1	Người dùng nhập name	
2	Người dùng nhập email	
3	Người dùng click “Choose file”	
5	Người dùng nhấn vào 1 ảnh	
6	Người dùng nhấn vào button “Open”	
7	Người dùng nhấn update	Hệ thống tạo một tài khoản mới với thông tin đầu vào và vai trò người dùng là khách hàng. Hệ thống chuyển hướng đến trang đăng nhập.

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
--------	--------------	-----------------	--

1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
2	Người dùng nhập thông tin và không chọn avatar mà nhấn button “update”	Hệ thống thông báo “Please fill out this field” tại khung nhập thông tin trống đó	
3	Người dùng nhập email không đúng định dạng, ví dụ ‘it.me’	Hệ thống thông báo “Please include an ‘@’ in the email address. ‘it.me002’ is missing an ‘@’”	
4	Người dùng nhập thông tin và không chọn avatar mà nhấn button “update”	Hệ thống thông báo “Internal Server Error”	
5	Người dùng chọn avatar không đúng định dạng file ảnh	Hệ thống hiển thị thông báo “User validation failed: Unsupported Zip File”	

Business Rules:

- Trường email phải đúng định dạng.
- Avatar phải đúng định dạng Image File.

Table 16: Đặc tả usecase Cập nhật thông tin cá nhân

2.13. Update Password

USE CASE – UC_13			
Use Case No.	UC_13	Use Case Version	1.0

Use Case Name	Update Password		
Author	Nguyen Quang Vu		
Date	23/11/2021	Priority	High

Actor: Customer

Summary: Cho phép người dùng cập nhật mật khẩu mới cho tài khoản.

Goal: Người dùng có thể sử dụng tài khoản của mình với mật khẩu mới.

Triggers: Người dùng nhấp vào icon change password để cập nhật mật khẩu..

Preconditions: người dùng phải có password tạm thời được gửi mail từ hệ thống.

Post conditions: Thành công: Đã cài lại mật khẩu mới.

Thất bại: Hệ thống hiển thị thông báo lỗi.

Step:

Step	Actor Action	System Response
1	Người dùng nhấn “change password”	Hiển thị update profile
2	Người dùng nhập password cũ	
3	Người dùng nhập password mới	
4	Người dùng nhập password xác nhận lại	

		Hệ thống xác nhận mật khẩu mới được đổi thành công
5	Người dùng nhấn "change"	

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	

1		Hệ thống hiển thị thông báo "Vui lòng kiểm tra kết nối internet của bạn"	

2	Người dùng không nhập thông tin mà nhấn button “Change”	Hệ thống thông báo “Please fill out this field” tại khung nhập thông tin trống đó	
3	Người dùng nhập password không đủ 8 ký tự	Hệ thống hiển thị thông báo “User validation failed: Password: Password should have more than 8 characters”	
4	Người dùng nhập password xác nhận không khớp với password	Hệ thống hiển thị thông báo “User validation failed: Password: Confirm Password is different from Password”	

Business Rules:

- Mật khẩu: độ dài của mật khẩu tối thiểu 8 ký tự.
- Hệ thống phải đảm bảo rằng không ai có thể đọc được mật khẩu của người dùng.
- Mật khẩu phải được mã hóa trong cơ sở dữ liệu.

Table 16: Đặc tả usecase Cập nhật mật khẩu

2.14. Create A Product Review

USE CASE – UC_14			
Use Case No.	UC_14	Use Case Version	1.0
Use Case Name	Create A Product Review		
Author	Nguyen Quang Vu		
Date	23/11/2021	Priority	High

Actor: Customer

Summary: Cho phép người dùng đánh giá trên mỗi sản phẩm.

Goal: Người dùng có thể xem những review sản phẩm trên hệ thống.

Triggers: Người dùng gửi đánh giá sản phẩm bằng cách nhấp vào “Submit” tại Submit Review.

Preconditions: None.

Post conditions: Thành công: Màn hình hiển thị đánh giá trên mỗi sản phẩm.

Thất bại: Màn hình không hiển thị đánh giá.

Step:

Step	Actor Action	System Response
1	Người dùng cuộn chuột xuống	
2	Người dùng chọn một sản phẩm bất kì	Hệ thống chuyển hướng đến trang chi tiết sản phẩm
3	Người dùng chọn button “Submit Review”	
4	Người dùng chọn sao và bình luận	
5	Người dùng chọn “Submit”	

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	

Business Rules:

- None.

*Table 17: Đặc tả usecase Tạo đánh giá***2.15. Add to Cart**

USE CASE – UC_15			
Use Case No.	UC_15	Use Case Version	1.0
Use Case Name	Add to Cart		
Author	Nguyen Quang Vu		
Date	23/11/2021	Priority	High

Actor: Customer

Summary: Cho phép người dùng thêm sản phẩm vào giỏ hàng.

Goal: Người dùng có thể thêm những sản phẩm vào giỏ hàng.

Triggers: Người thêm sản phẩm bằng cách nhấn vào “Add to Cart”.

Preconditions: None.

Post conditions: Thành công: Giỏ hàng hiển thị sản phẩm người dùng đã thêm.

Thất bại: Màn hình không hiển thị sản phẩm trong giỏ hàng.

Step:

Step	Actor Action	System Response
1	Người dùng cuộn chuột xuống	
2	Người dùng chọn một sản phẩm bất kì	Hệ thống chuyển hướng đến trang chi tiết sản phẩm
3	Người dùng chọn button “Add to Cart”	Hệ thống thông báo “Item added to cart”

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 18: Đặc tả usecase Thêm sản phẩm vào Giỏ hàng

2.16. Take an order

USE CASE – UC_16			
Use Case No.	UC_16	Use Case Version	1.0
Use Case Name	Take an order		
Author	Nguyen Quang Vu		
Date	23/11/2021	Priority	High

Actor: Customer

Summary: Cho phép người dùng đặt sản phẩm có trong giỏ hàng.

Goal: Người dùng có thể xem những sản phẩm trong giỏ hàng.

Triggers: Người dùng xem sản phẩm trong giỏ hàng bằng cách nhấp vào “Cart” trên menu.

Preconditions: None.

Post conditions: Thành công: Màn hình hiển thị giỏ hàng.

Thất bại: Màn hình không hiển thị giỏ hàng.

Step:

Step	Actor Action	System Response
1	Người dùng trỏ chuột vào avatar để xuất hiện menu	
2	Người dùng chọn “cart”	Hệ thống chuyển hướng đến trang giỏ hàng của sản phẩm

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	

	1		
		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	

Business Rules:

- None.

Table 19: Đặc tả usecase Đặt hàng

2.17. Check out

USE CASE – UC_17			
Use Case No.	UC_17	Use Case Version	1.0
Use Case Name	Check out		
Author	Nguyen Quang Vu		
Date	23/11/2021	Priority	High

Actor: Customer

Summary: Cho phép khác hàng thanh toán sản phẩm có trong giỏ hàng.

Goal: Khách hàng có thể xem những sản phẩm trong giỏ hàng.

Triggers: Người dùng thanh toán sản phẩm trong giỏ hàng bằng cách nhấn vào “check out” trong giỏ hàng.

Preconditions: None.

Post conditions: Thành công: Màn hình hiển thị trang thanh toán.

Thất bại: Màn hình không hiển thị trang thanh toán.

Step:

Step	Actor Action	System Response
1	Người dùng click vào “check out” trên mỗi sản phẩm cần thanh toán trong giỏ hàng	Hiển thị màn hình Shipping Details
2	Người dùng nhập address	
3	Người dùng nhập city	
4	Người dùng nhập pin code	
5	Người dùng nhập phone number	
6	Người dùng chọn quốc gia	
7	Người dùng chọn tỉnh	
8	Người dùng nhấn “continue”	Hệ thống chuyển sang giao diện Confirm Order
9	Người dùng click “Proceed to payment”	Hệ thống chuyển sang giao diện Card Information

10	Người dùng nhập thông tin thẻ ngân hàng	
11	Người dùng nhấn “pay...”	Hệ thống xuất hiện trang thanh toán để minh xác nhận
12	Người dùng nhấn “complete authentication”	Hệ thống xuất hiện “Your Order has been Placed successfully”

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	

1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
---	--	--	--

2	Người dùng không nhập 1 trong 4 “address, city, pincode, number”	Hệ thống thông báo “Please fill out this field”	
3	Người dùng nhập số điện thoại không đủ 10 số	Hệ thống thông báo “Phone number should be 10 digits long”	
4	Người dùng không chọn country	Hệ thống không chuyên trang	
5	Người dùng không nhập số thẻ, ngày tháng, cvv	Hệ thống thông báo “your card number is incomplet”	

Business Rules:

- None.
- Số điện thoại nhập đủ 10 số
- Nhập đầy đủ thông tin address, city, pincode, number
- Chọn country , state
- Nhập đầy đủ số thẻ, ngày tháng, cvv

Table 20: *Đặc tả usecase Thanh toán*

2.18. See all orders

USE CASE – UC_18			
Use Case No.	UC_18	Use Case Version	1.0
Use Case Name	See all orders		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Customer

Summary: Cho phép xem những sản phẩm đã đặt trong order.

Goal: Người dùng có thể xem những sản phẩm trong trang đặt hàng.

Triggers: Người dùng xem sản phẩm đã đặt trong order bằng cách nhấn vào “Orders” trên menu.

Preconditions: None.

Post conditions: Thành công: Màn hình hiển thị trang đặt hàng.

Thất bại: Màn hình không hiển thị trang đặt hàng.

Step:

Step	Actor Action	System Response
1	Người dùng trỏ chuột vào logo trang cá nhân góc bên phải	
2	Người dùng chọn “orders” trên thanh menu	Hệ thống chuyển hướng đến trang những sản phẩm Đã đặt hàng
3	Người dùng chọn “action” trên mỗi sản phẩm	Hệ thống chuyển hướng đến trang chi tiết đơn hàng

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 21: Đặc tả usecase Xem tất cả các đơn đặt hàng

2.19. See a single order

USE CASE – UC_18			
Use Case No.	UC_19	Use Case Version	1.0
Use Case Name	See a single orders		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Customer

Summary: Cho phép xem những chi tiết 1 sản phẩm đã đặt trong order.

Goal: Người dùng có thể xem 1 sản phẩm trong trang chi tiết đặt hàng.

Triggers: Người dùng xem chi tiết 1 sản phẩm đã đặt trong order bằng cách nhấp vào “Action” mỗi sản phẩm trong trang orders.

Preconditions: None.

Post conditions: Thành công: Màn hình hiển thị trang chi tiết đơn hàng.

Thất bại: Màn hình không hiển thị trang chi tiết đơn hàng.

Step:

Step	Actor Action	System Response
2	Người dùng chọn icon hình vuông của thanh “actions” mỗi sản phẩm	Hệ thống chuyển hướng đến trang chi tiết sản phẩm đã đặt hàng
3	Người dùng chọn logo sản phẩm trên mỗi trang chi tiết sản phẩm	Hệ thống chuyển hướng đến trang chi tiết sản phẩm

Alternative Scenario: N/A

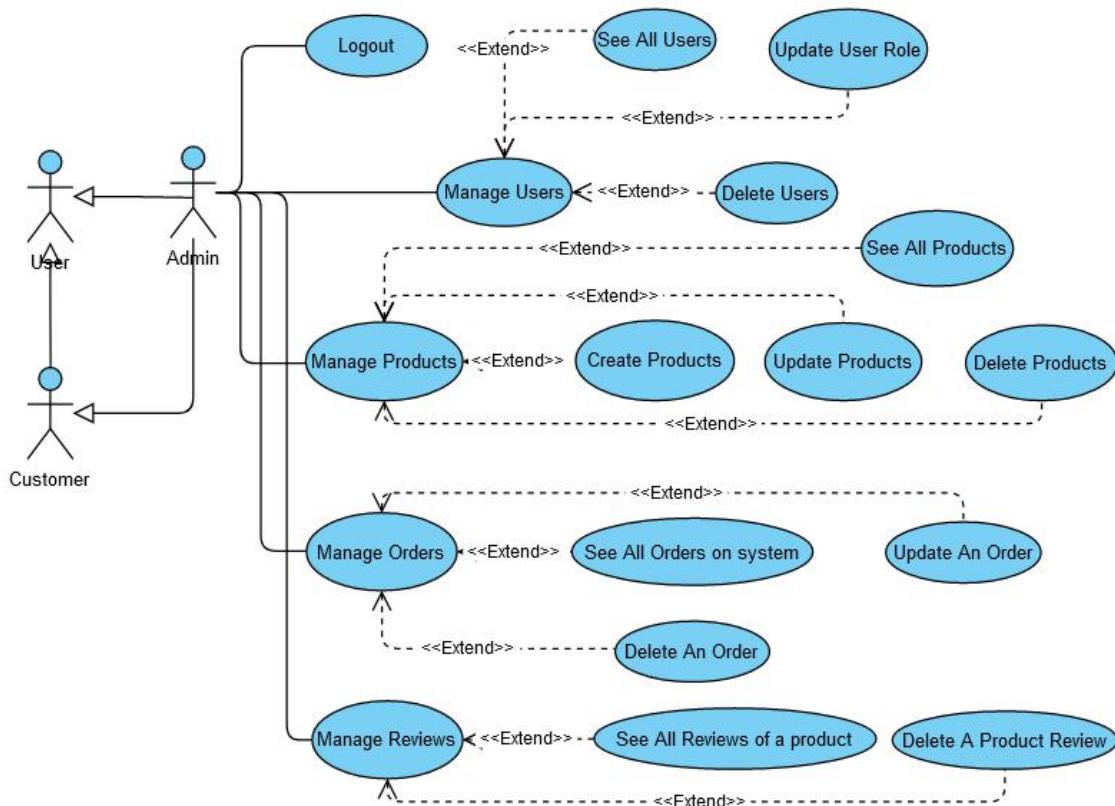
Exceptions:

N o	Actor Action	System Response	
1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	

Business Rules:

- None.

Table 22: Đặc tả usecase Xem thông tin chi tiết 1 đơn đặt hàng



Hình 4: Usecases của Actor Admin

Figure 3.2.3: Admin

2.20. See All Users

USE CASE – UC_20			
Use Case No.	UC_20	Use Case Version	1.0

Use Case Name	See All Users		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép admin xem danh sách tất cả các người dùng có trên hệ thống.

Goal: Admin có thể xem danh sách tất cả các người dùng có trên hệ thống để thực hiện các chức năng quản lý người dùng.

Triggers: Admin xem danh sách người dùng trên hệ thống bằng cách nhấp vào “Users” trên thanh sidebar bên trái của Dashboard Admin.

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình hiển thị danh sách tất cả các người dùng trên hệ thống.

Thất bại: Màn hình hiển thị trang loading.

Step:

Step	Actor Action	System Response
1	Admin chọn icon Dashboard trên icon Avatar	Hệ thống chuyển hướng đến trang Dashboard Admin
2	Admin chọn “Users” trên thanh sidebar bên trái Dashboard Admin	Hệ thống chuyển hướng đến trang Manage Users Màn hình hiển thị danh sách tất cả các user có trên hệ thống

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	

1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
Business Rules:			
<ul style="list-style-type: none"> • None. 			

Table 23: Đặc tả usecase Xem tất cả các người dùng trên hệ thống

2.21. Update User Role

USE CASE – UC_21			
Use Case No.	UC_21	Use Case Version	1.0
Use Case Name	Update User Role		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép admin nâng cấp quyền hạn của người dùng khác.

Goal: Admin có thể nâng cấp quyền hạn của người dùng khác từ Khách Hàng thành Quản Trị Viên.

Triggers: Admin nâng cấp quyền hạn của người dùng khác bằng cách chọn icon Pencil của 1 người dùng trên danh sách tất cả các người dùng trên hệ thống.

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Hệ thống cập nhật quyền hạn của người dùng vừa được admin nâng cấp.

Thất bại: Màn hình hiển thị trang loading.

Step:

Step	Actor Action	System Response
1	Admin chọn icon Pencil của 1 người dùng trên danh sách tất cả các người dùng trên hệ thống	Hệ thống chuyển hướng đến trang Update User Role
2	Admin chọn quyền hạn mới của người dùng đó trên danh sách quyền hạn	
2	Admin chọn button “Update”	Hệ thống chuyển hướng đến trang danh sách tất cả các người dùng trên hệ thống Màn hình hiển thị quyền hạn của người dùng vừa được nâng cấp lên thành quản trị viên

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 24: Đặc tả usecase Nâng cấp quyền hạn người dùng

2.22. Delete Users

USE CASE – UC_22			
Use Case No.	UC_22	Use Case Version	1.0
Use Case Name	Delete Users		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép admin xóa người dùng trên hệ thống.

Goal: Admin có thể xóa người dùng trên hệ thống.

Triggers: Admin xóa người dùng trên hệ thống bằng cách chọn icon Bin của 1 người dùng trên danh sách tất cả các người dùng trên hệ thống.

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Hệ thống xóa vĩnh viễn tài khoản người dùng vừa được admin chọn xóa.

Thất bại: Màn hình hiển thị trang loading.

Step:

Step	Actor Action	System Response
1	Admin chọn icon Bin của 1 người dùng trên danh sách tất cả các người dùng trên hệ thống	Hệ thống xóa vĩnh viễn tài khoản người dùng vừa được admin chọn. Màn hình hiển thị danh sách tất cả các người dùng (không bao gồm người dùng vừa được xóa)

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 25: Đặc tả usecase Xóa người dùng

2.23. See All Products

USE CASE – UC_23			
Use Case No.	UC_23	Use Case Version	1.0
Use Case Name	See All Products		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép admin xem danh sách tất cả các sản phẩm có trên hệ thống.

Goal: Admin có thể xem danh sách tất cả các sản phẩm có trên hệ thống để thực hiện các chức năng quản lý sản phẩm.

Triggers: Admin xem danh sách sản phẩm trên hệ thống bằng cách nhấp vào “Products - All” trên thanh sidebar bên trái của Dashboard Admin.

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình hiển thị danh sách tất cả các sản phẩm trên hệ thống.

Thất bại: Màn hình hiển thị trang loading.

Step:

Step	Actor Action	System Response
1	Admin chọn icon Dashboard trên icon Avatar	Hệ thống chuyển hướng đến trang Dashboard Admin
2	Admin chọn “Products” trên thanh sidebar bên trái Dashboard Admin	
3	Admin chọn “All” trên content mở rộng của “Products”	Hệ thống chuyển hướng đến trang Manage Products Màn hình hiển thị danh sách tất cả các sản phẩm có trên hệ thống

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 26: Đặc tả usecase Xem tất cả các sản phẩm có trên hệ thống

2.24. Create Products

USE CASE – UC_24			
Use Case No.	UC_24	Use Case Version	1.0
Use Case Name	Create Products		
Author	Bùi Hà Nhi		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép admin tạo mới 1 sản phẩm.

Goal: Admin có thể tạo mới 1 sản phẩm.

Triggers: Admin tạo mới 1 sản phẩm bằng cách chọn “Products - Create” trên thanh sidebar bên trái của Dashboard Admin.

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Hệ thống cập nhật sản phẩm mới lên danh sách.

Thất bại: Màn hình hiển thị trang loading.

Step:

Step	Actor Action	System Response
1	Admin chọn icon Dashboard trên icon Avatar	Hệ thống chuyển hướng đến trang Dashboard Admin
2	Admin chọn “Products” trên thanh sidebar bên trái Dashboard Admin	
3	Admin chọn “Create” trên content mở rộng của “Products”	Hệ thống chuyển hướng đến trang Create Products Màn hình hiển thị form tạo mới 1 sản phẩm
4	Admin nhập tên sản phẩm trong khung “Product Name”	
5	Admin nhập giá sản phẩm trong khung “Price”	
6	Admin nhập mô tả sản phẩm trong khung “Product Description”	
7	Admin chọn danh mục của sản phẩm trong combobox “Choose Category”	Trường danh mục hiển thị ra 1 list các loại danh mục sản phẩm cho Admin lựa chọn
8	Admin nhập số lượng sản phẩm trong khung “Stock”	
9	Admin nhấn button “Browse...”	Hệ thống tự chuyển tiếp mở 1 cửa sổ File Explorer cho Admin chọn ảnh sản phẩm

10	Admin nhấn button “CREATE”	<p>Hệ thống chuyển tiếp về trang danh sách tất cả các sản phẩm</p> <p>Màn hình hiển thị trang danh sách tất cả các sản phẩm</p> <p>Sản phẩm mới vừa được admin tạo được đưa vào cơ sở dữ liệu</p>
----	----------------------------	---

Alternative Scenario: N/A

Exceptions:

	Actor Action	System Response	
N o			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

2	Admin không nhập thông tin sản phẩm và không chọn ảnh sản phẩm mà nhấn button “CREATE”	Hệ thống thông báo “Please fill out this field” tại khung nhập thông tin trống đó	
3	Admin chọn ảnh sản phẩm không đúng định dạng file ảnh	Hệ thống hiển thị thông báo “User validation failed: Unsupported Zip File”	
4	Admin nhập thông tin sản phẩm và không chọn ảnh sản phẩm mà nhấn button “CREATE”	Hệ thống không hiển thị thông báo, màn hình vẫn giữ trạng thái đang tạo mới sản phẩm	
5	Admin nhập giá sản phẩm không phải là số	Hệ thống thông báo “Please enter a number” tại khung nhập giá sản phẩm	
6	Admin nhập giá sản phẩm có nhiều hơn 9 ký tự số	Hệ thống thông báo “Price cannot exceed 9 characters” tại khung nhập giá sản phẩm	
7	Admin không nhập số lượng sản phẩm	Hệ thống mặc định số lượng sản phẩm đó là 1 trong kho. Sản phẩm được Admin tạo thành công	
8	Admin nhập số lượng sản phẩm không phải là số	Hệ thống thông báo “Please enter a number” tại khung nhập số lượng sản phẩm	
9	Admin nhập số lượng sản phẩm nhiều hơn 4 ký tự	Hệ thống thông báo “Stock cannot exceed 4 characters” tại khung nhập số lượng sản phẩm	

Business Rules:

- Avatar phải đúng định dạng Image File.
- Giá sản phẩm và số lượng sản phẩm phải là số
- Giá sản phẩm không được vượt quá 9 ký tự số
- Số lượng sản phẩm không được vượt quá 4 ký tự số

Table 27: Đặc tả usecase Tạo mới 1 sản phẩm

2.25. Update Products

USE CASE – UC_25			
Use Case No.	UC_25	Use Case Version	1.0
Use Case Name	Update Products		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép cập nhật thông tin 1 sản phẩm.

Goal: Admin có thể cập nhật thông tin 1 sản phẩm trong trang Update Product.

Triggers: Admin cập nhật chi tiết 1 sản phẩm bằng cách nhấp vào cây bút chì trên mỗi sản phẩm trong trang all products.

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình hiển thị cập nhật sản phẩm thành công.

Thất bại: Màn hình không hiển thị thông tin chi tiết sản phẩm được cập nhật.

Step:

Step	Actor Action	System Response
2	Admin nhấp vào cây bút chì trên mỗi sản phẩm trong trang all products.	Hệ thống chuyển hướng đến trang Update Product

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 28: Đặc tả usecase Cập nhật 1 sản phẩm

2.26. Delete Products

USE CASE – UC_26			
Use Case No.	UC_26	Use Case Version	1.0
Use Case Name	Delete Products		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép xóa 1 sản phẩm.

Goal: Admin có thể xóa 1 sản phẩm trong trang All Product.

Triggers: Admin xóa 1 sản phẩm bằng cách nhấp vào icon thùng rác trên mỗi sản phẩm trong cột Actions

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình không còn hiển thị sản phẩm vừa xóa.

Thất bại: Màn hình vẫn còn thông tin sản phẩm vừa xóa.

Step:

Step	Actor Action	System Response
2	Admin nhấp vào icon thùng rác trên mỗi sản phẩm trong cột “Actions”	Hệ thống cập nhật lại trang Update Product

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 29: Đặc tả usecase Xóa 1 sản phẩm

2.27. See All Orders on system

USE CASE – UC_27			
Use Case No.	UC_27	Use Case Version	1.0
Use Case Name	See All Orders on system		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép xem chi tiết những sản phẩm mà khách hàng đã đặt.

Goal: Admin có thể xem chi tiết những sản phẩm khách hàng đã đặt trong All Orders

Triggers: Admin xem những sản phẩm đã đặt bằng cách nhấp vào Orders trong Dashboard

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình hiển thị những sản phẩm khách hàng đặt trong All Orders.

Thất bại: Màn hình loading.

Step:

Step	Actor Action	System Response
1	Admin nhấp vào Orders trong Dashboard	Hệ thống chuyển hướng sang trang All Orders
2	Admin nhấp vào cây bút trên mỗi sản phẩm để xem chi tiết mỗi đơn hàng	Hệ thống chuyển hướng sang trang thông tin chi tiết đơn hàng
3	Admin nhấp vào thùng rác cột actions để xóa 1 đơn hàng	

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 30: Đặc tả usecase Xem các đơn hàng trên hệ thống

2.28. Update An Order

USE CASE – UC_28			
Use Case No.	UC_28	Use Case Version	1.0
Use Case Name	Update An Order		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép xem và cập nhật thông tin vận chuyển mà khách hàng đã đặt.

Goal: Admin có thể xem và cập nhật thông tin vận chuyển mà khách hàng đã đặt

Triggers: Admin xem và cập nhật thông tin vận chuyển bằng cách nhấp vào cây bút trên cột Actions

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình hiển thị thông tin chi tiết những sản phẩm khách hàng đặt.

Thất bại: Màn hình vẫn còn ở trang All Orders.

Step:

Step	Actor Action	System Response
1	Admin nhấp vào Orders trong Dashboard	Hệ thống chuyển hướng sang trang All Orders
2	Admin nhấp vào cây bút trên mỗi sản phẩm để xem chi tiết mỗi đơn hàng	Hệ thống chuyển hướng sang trang thông tin chi tiết đơn hàng
3	Admin nhấp thay đổi thông tin vận chuyển trên Process Order	

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 31: Đặc tả usecase Cập nhật trạng thái đơn hàng trên hệ thống

2.29. Delete An Order

USE CASE – UC_29			
Use Case No.	UC_29	Use Case Version	1.0
Use Case Name	Delete An Order		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép xóa 1 sản phẩm mà khách hàng đã đặt.

Goal: Admin có thể xem và xóa sản phẩm mà khách hàng đã đặt

Triggers: Admin xem xóa sản phẩm order bằng cách nhấp vào thùng rác trên cột Actions

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình không còn hiển thị thông tin sản phẩm mà admin vừa xóa.

Thất bại: Màn hình vẫn còn ở trang All Orders.

Step:

Step	Actor Action	System Response
1	Admin nhấp vào thùng rác trên cột Actions	Hệ thống cập nhật lại trang

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

- None.

Table 32: Đặc tả usecase Xóa đơn hàng trên hệ thống

2.30. See All Reviews of a Product

USE CASE – UC_30			
Use Case No.	UC_30	Use Case Version	1.0
Use Case Name	See All Reviews of a Product		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép xem tất cả đánh giá trên từng sản phẩm.

Goal: Admin có thể xem tất cả đánh giá trên từng sản phẩm

Triggers: Admin xem tất cả đánh giá trên từng sản phẩm bằng cách nhấp vào Reviews trên Dashboard

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình hiển thị trang ALL REVIEWS

Thất bại: Màn hình vẫn còn ở trang Dashboard.

Step:

Step	Actor Action	System Response
1	Admin nhấp vào Reviews trên Dashboard	Hệ thống chuyển sang trang All Reviews
2	Admin gõ id của sản phẩm	Hệ thống chuyển hướng sang trang chi tiết sản phẩm

Alternative Scenario: N/A

Exceptions:

No	Actor Action	System Response	
1		Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	

2	Không nhập id của sản phẩm	Trang không load thông tin của sản phẩm	
3	Nhập sai id sản phẩm	Trang không load thông tin của sản phẩm	

Business Rules:

- Nhập chính xác id của sản phẩm

Table 33: *Đặc tả usecase Xem tất cả các đánh giá của 1 sản phẩm trên hệ thống*

2.31. Delete A Product Review

USE CASE – UC_31			
Use Case No.	UC_31	Use Case Version	1.0
Use Case Name	Delete A Product Review		
Author	Nguyen Quang Vu		
Date	24/11/2021	Priority	High

Actor: Admin

Summary: Cho phép xóa tất cả đánh giá trên mỗi sản phẩm.

Goal: Admin có thể xóa tất cả đánh giá trên mỗi sản phẩm

Triggers: Admin xóa tất cả đánh giá trên từng sản phẩm bằng cách nhấn vào thùng rác trên cột Actions

Preconditions: Người dùng phải đăng nhập thành công với quyền hạn Admin của hệ thống.

Post conditions: Thành công: Màn hình cập nhật lại trang All Review không còn đánh giá đó nữa

Thất bại: Màn hình vẫn còn đánh giá

Step:

Step	Actor Action	System Response
1	Admin nhấn vào thùng rác trên cột Actions	Hệ thống cập nhật lại trang All Review

Alternative Scenario: N/A

Exceptions:

N o	Actor Action	System Response	
1			

	1	Trình duyệt hiển thị thông báo “Hmm. We’re having trouble finding that site.” khi internet bị mất.	
--	---	--	--

Business Rules:

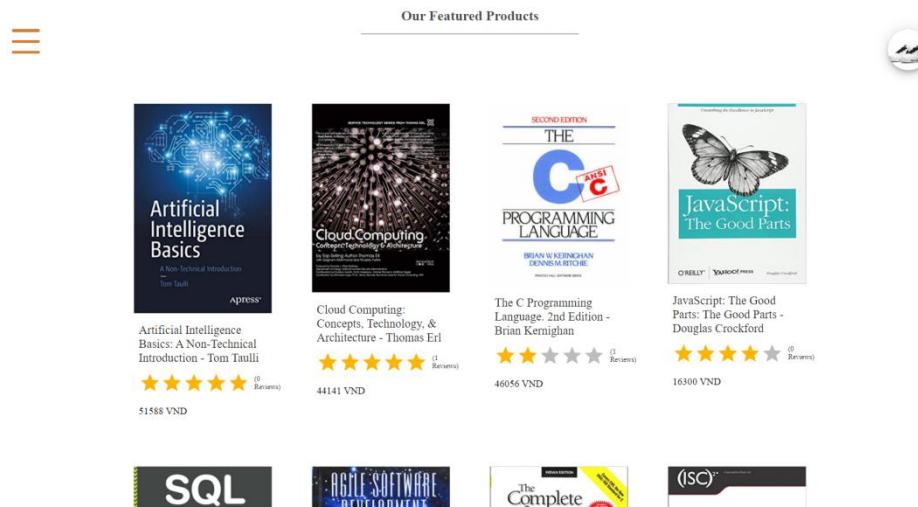
- Nhập chính xác id của sản phẩm

Table 34: Đặc tả usecase Xóa các đánh giá của 1 sản phẩm trên hệ thống

CHƯƠNG IV: GIAO DIỆN - HƯỚNG DẪN SỬ DỤNG

1. Giao diện trang chủ

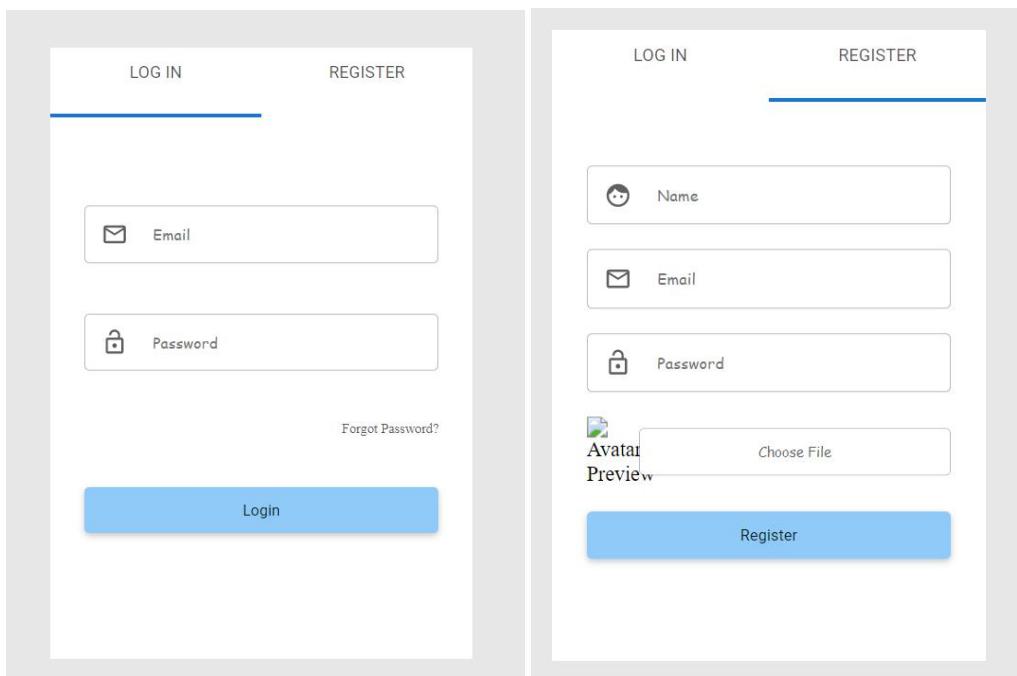
Trang chủ sẽ chứa thông tin, hình ảnh về các sản phẩm nổi bật. Đồng thời sẽ hiển thị ra thông tin 8 sản phẩm nổi bật nhất. Giao diện trang chủ như hình 4.1.1



Hình 4.1.1 Giao diện trang chủ

2. Trang đăng nhập, đăng ký

Để thực hiện việc quản lý các đơn đặt hàng đã đặt mua một cách dễ dàng, khách hàng có thể đăng ký, đăng nhập.



Hình 4.2.1 Giao diện trang đăng nhập, đăng ký

3. Giao diện trang sản phẩm theo danh mục sản phẩm

Hiện tại website cung cấp 6 danh mục chính: Science Technology Books, Comics, Novels, Foreign Language Learning Books, Course Vouchers, School Supplies.

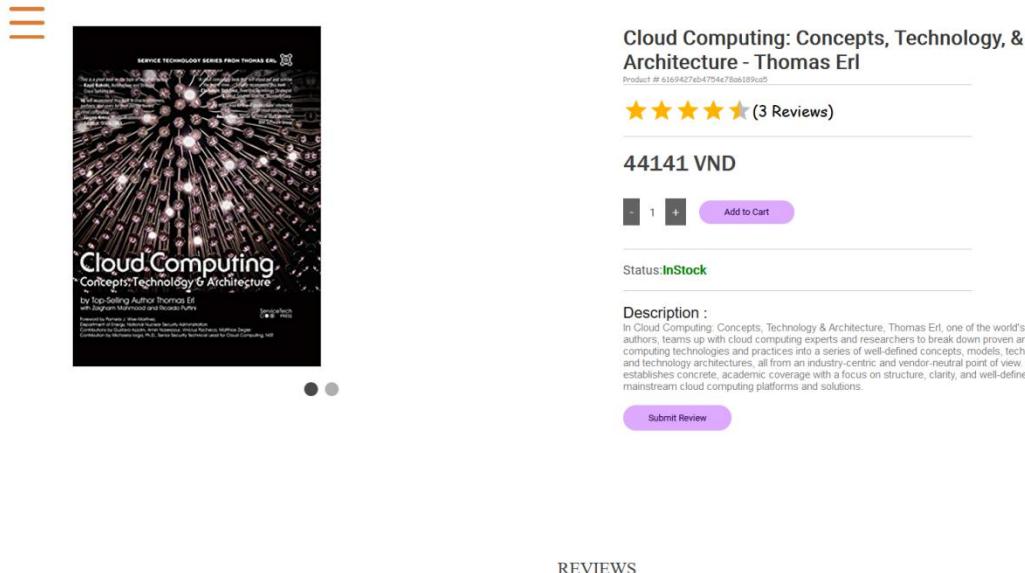
Product	Price	Rating	Reviews
Artificial Intelligence Basics	51588 VND	5 stars	(0 Reviews)
Cloud Computing: Concepts, Technology, & Architecture	44141 VND	5 stars	(1 Review)
THE C PROGRAMMING LANGUAGE, SECOND EDITION	46056 VND	5 stars	(1 Review)
JavaScript: The Good Parts	16300 VND	5 stars	(0 Reviews)



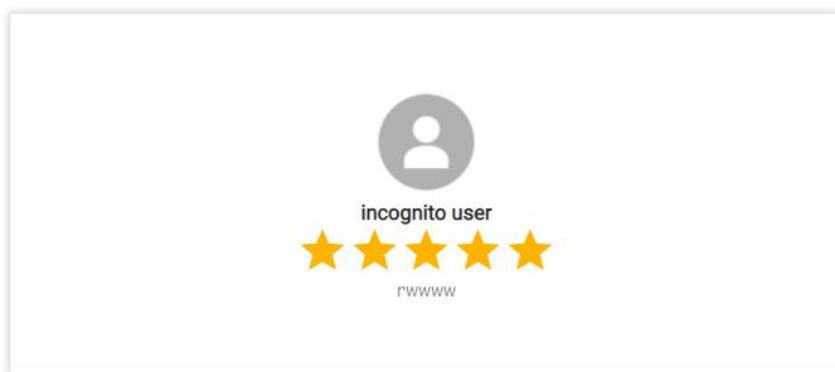
Hình 4.3.1 Giao diện trang sản phẩm theo danh mục sản phẩm

4. Giao diện trang chi tiết sản phẩm

Trang chi tiết sản phẩm chứa thông tin chi tiết về sản phẩm. Ngoài ra, có thể nhận xét, đánh giá về sản phẩm.



Hình 4.4.1 Giao diện trang chi tiết sản phẩm

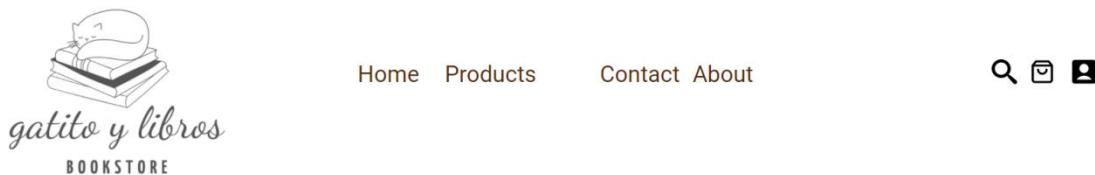


Hình 4.4.2 Giao diện trang đánh giá, review

5. Giỏ hàng

Để thực hiện việc mua một sản phẩm. Có ba bước:

- Click add to card ở giao diện chi tiết sản phẩm mục 4.4.1
- Click trực tiếp vào hình giỏ hàng trang header như hình dưới đây



Hình 4.5.1 Giao diện header

- Sau khi thêm sản phẩm vào giỏ hàng. Khách hàng có thể click vào giỏ hàng kiểm tra giỏ hàng, cập nhật và xác nhận giỏ hàng.

Products	Quantities	Subtotal Price
Cloud Computing: Concepts, Technology, & Architecture - Thomas Erl Price: 44141 VND Remove	[-] [+]	44141 VND
JavaScript: The Good Parts - Douglas Crockford Price: 16300 VND Remove	[-] [+]	16300 VND
Agile Software Development: Principles, Patterns, and Practices - Robert C. Martin Price: 55471 VND Remove	[-] [+]	55471 VND
<hr/>		Gross Total
		115912 VND

4.2.5.2 Giao diện giỏ hàng

6. Thông tin mua hàng, xác nhận thanh toán

Sau khi xem giỏ hàng, nếu khách hàng muốn mua sản phẩm đó. Người dùng bắt buộc phải cung cấp thông tin để mua hàng. Nếu người dùng đã có tài khoản. Chỉ cần đăng nhập. Các trường thông tin mua hàng sẽ tự động được cập nhật. Trong trường hợp người dùng chưa có tài khoản, khách hàng buộc phải đăng ký

The screenshot shows a shipping details form with the following fields:

- Address: Tây Ninh
- City: Quy Nhơn
- Postal Code: 10000
- Phone Number: 0123456788
- Country: Vietnam
- Province: Ho Chi Minh City

A blue "Continue" button is located at the bottom of the form.

Hình 4.6.1 Giao diện trang điền thông tin người mua hàng

Sau khi hoàn tất các thông tin mua hàng, website sẽ chuyển khách hàng đến trang thanh toán. Thanh toán trực tuyến qua Stripe.

Shipping Information

Name: Quang Vu
Phone: 0123456788
Address: Tây Ninh, Quy Nhơn, SG, 10000, VN

There're your Cart Items:

Image	Product Description	Quantity	Unit Price	Total Price
	Cloud Computing: Concepts, Technology, & Architecture - Thomas Erl	1	44141 VND	44141 VND
	JavaScript: The Good Parts: The Good Parts - Douglas Crockford	1	16300 VND	16300 VND
	Agile Software Development: Principles, Patterns, and Practices - Robert C. Martin	1	55471 VND	55471 VND

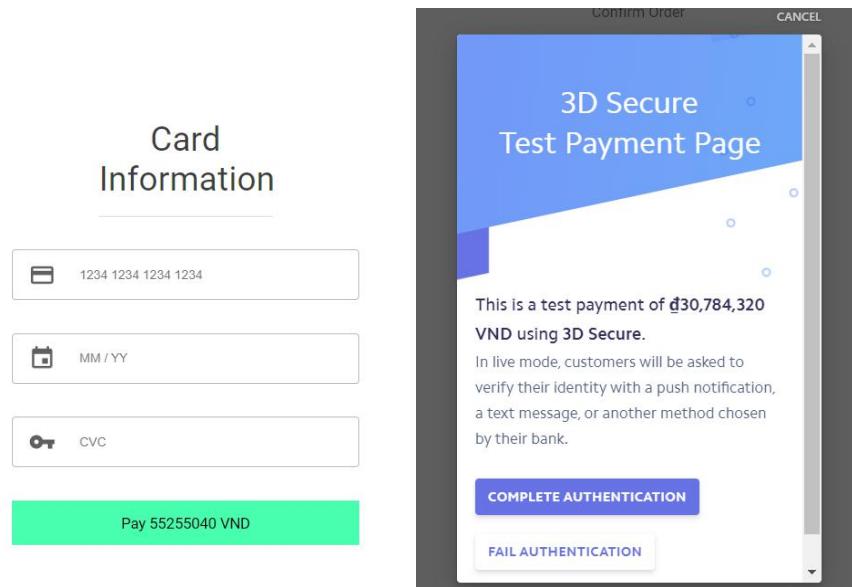
Order Summary

Item	Value
Subtotal:	115912 VND
Shipping Charges:	50000 VND
GST:	2781.888 VND
Total:	168693.888 VND

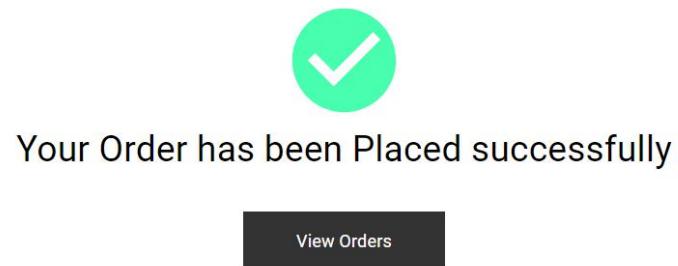
Proceed To Payment

Hình 4.6.2 Giao diện trang thanh toán

Sau khi tiến hành xác nhận thanh toán thành công, trang sẽ tự động điều khiển khách hàng về trang xác nhận đặt hàng thành công.



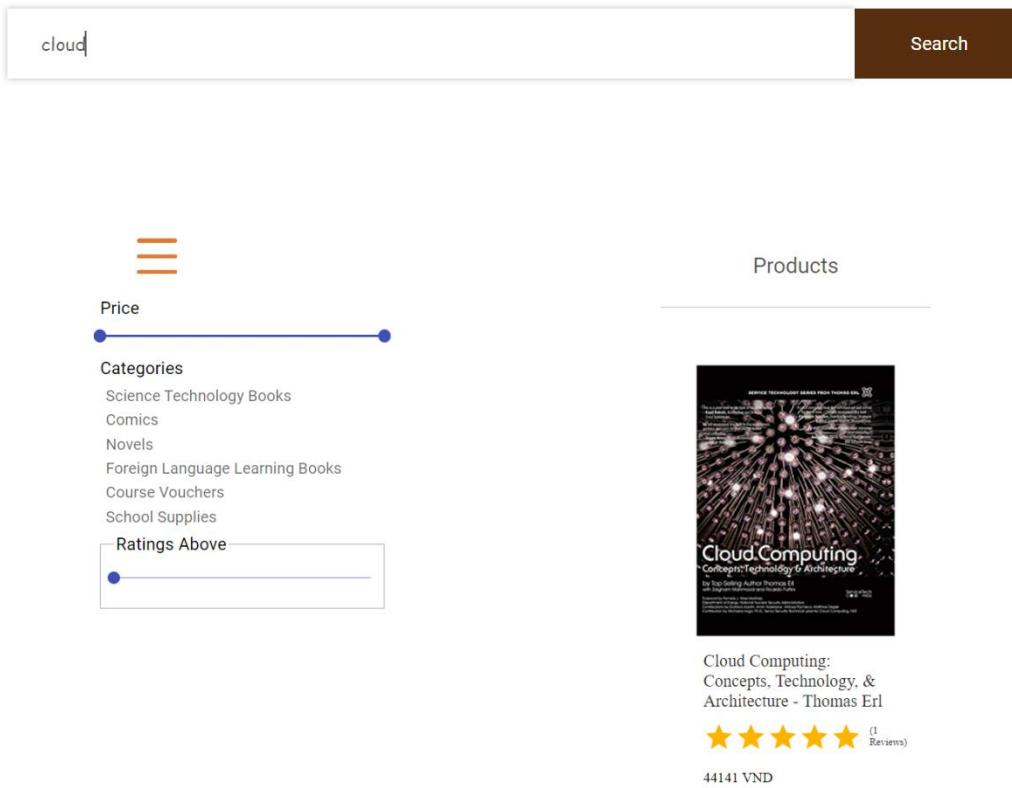
Hình 4.6.3 Giao diện trang thanh toán



Hình 4.6.4 Giao diện trang xác nhận thanh toán thành công

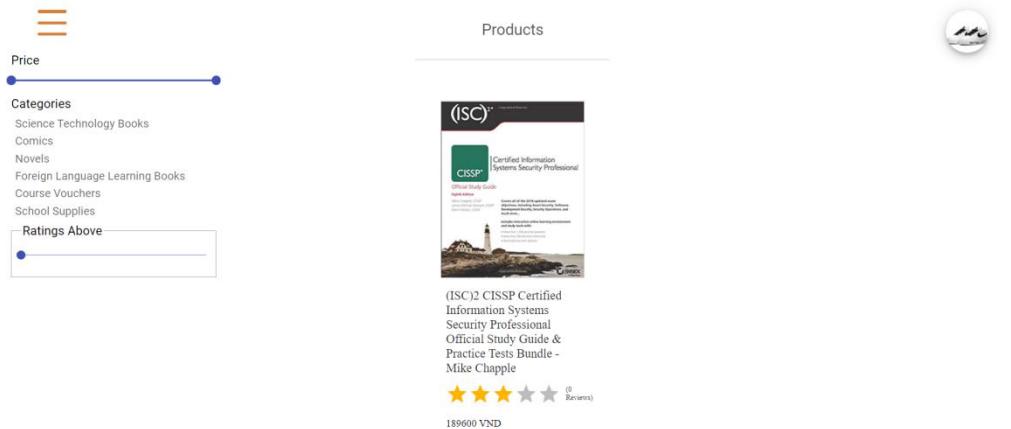
7. Tìm kiếm

Tìm kiếm theo tên



Hình 4.7.1 Giao diện tìm kiếm

8. Giao diện lọc theo giá hoặc review sản phẩm



Hình 4.8.1 Giao diện lọc theo giá



Hình 4.8.2 Giao diện lọc theo review

9. Liên hệ

Thông tin liên hệ



CONTACT: 18110168@STUDENT.HCMUTE.EDU.VN

CONTACT: 18110241@STUDENT.HCMUTE.EDU.VN



Hình 4.9 Giao diện trang liên hệ

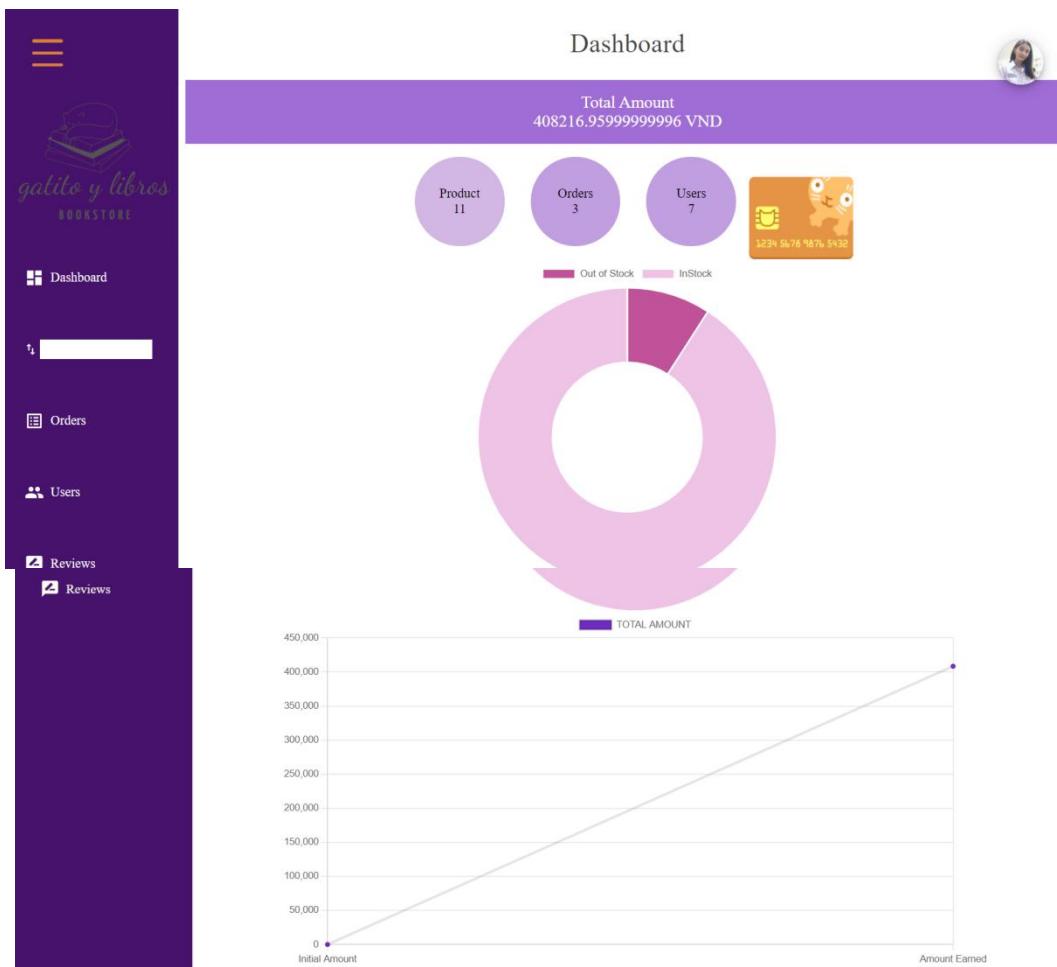
10. Giao diện và chức năng admin

Admin (Quản trị viên) là quyền hạn cao nhất trên hệ thống, có thể thực hiện được các chức năng của một user thường và các chức năng đặc quyền chỉ có thể quản trị viên mới có thể sử dụng:

- Quản lý sản phẩm:
 - + Xem tất cả sản phẩm trên hệ thống
 - + Sửa thông tin 1 sản phẩm
 - + Thêm số lượng 1 sản phẩm vào kho
 - + Thêm sản phẩm mới
 - + Xóa sản phẩm
- Quản lý đơn hàng:
 - + Xem tất cả các đơn hàng trên hệ thống
 - + Hủy đơn hàng

- + Sửa trạng thái đơn hàng
- Quản lý người dùng:
 - + Xem tất cả các người dùng trên hệ thống
 - + Nâng cấp quyền hạn tài khoản người dùng
 - + Xóa người dùng vĩnh viễn
- Quản lý đánh giá:
 - + Xem tất cả các đánh giá của người dùng trên 1 sản phẩm
 - + Xóa các đánh giá có nội dung không hợp lệ

Trên giao diện chính của Dashboard admin, có để tổng doanh thu trên hệ thống từ các đơn hàng, thống kê tổng số sản phẩm, đơn hàng, người dùng trên hệ thống, có biểu đồ phân tích sản phẩm còn trong kho hàng và sản phẩm hết hàng, có biểu đồ phân tích thu nhập ban đầu đến thu nhập hiện tại.



Hình 4.10.1. Giao diện chính trang quản trị viên

Product ID	Name	Stock	Price	A
6168669b56af38fcca86f9e6	Artificial Intelligence Basics: A Non-Technical Introduction - Tom ...	100	51,588	
6169427eb4754e78a6189ca5	Cloud Computing: Concepts, Technology, & Architecture - Thomas... ...	20	44,141	
61694320b4754e78a6189ca7	The C Programming Language: 2nd Edition - Brian Kernighan	20	46,056	
6169438db4754e78a6189ca9	JavaScript: The Good Parts: The Good Parts - Douglas Crockford	100	16,300	
616943d5b4754e78a6189cab	SQL QuickStart Guide: The Simplified Beginner's Guide to Managi... ...	10	15,462	
61694435b4754e78a6189cad	Agile Software Development: Principles, Patterns, and Practices -...	100	55,471	
616944aebe78a6189cb0	C++: The Complete Reference, 4th Editions - Herbert Schildt	10	84,608	
616c41e96208eba86f0ac233	(ISC)2 CISSP Certified Information Systems Security Professiona... ...	20	189,600	
616c42556208eba86f0ac235	Practical Packet Analysis: Using Wireshark to Solve Real-world N... ...	100	40,000	
616c428e6208eba86f0ac237	Black Hat Python, 2nd Edition: Python Programming for Hackers	10	30,000	

Hình 4.10.2. Giao diện danh sách sản phẩm trên hệ thống được quản lý bởi quản trị viên

Create Product

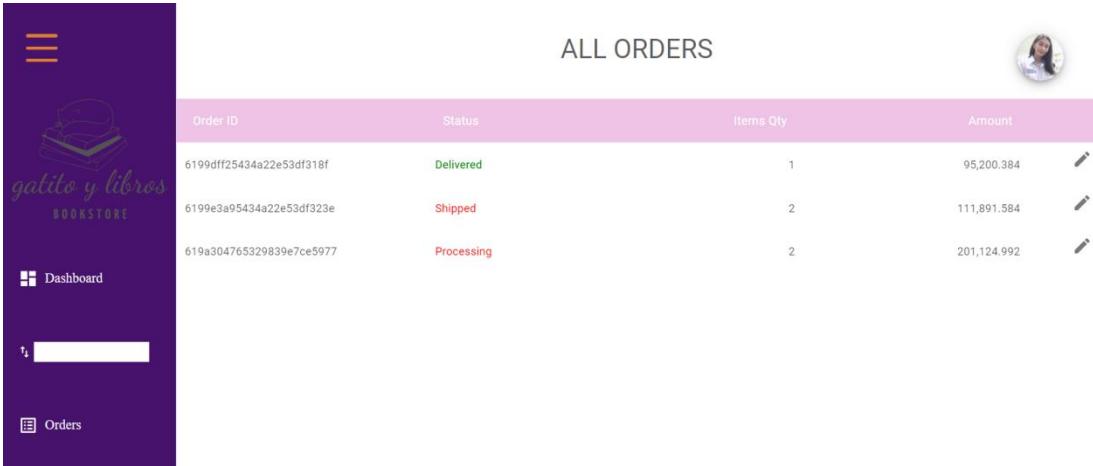
CREATE

Hình 4.10.3. Giao diện thêm sản phẩm mới được quản lý bởi quản trị viên

Update Product

A	Artificial Intelligence Basics: A Non-
\$	51588
File	Google, Amazon, Facebook, and similar tech giants are far from
Category	Science Technology Books
Stock	100
Choose Files	
	
CREATE	

Hình 4.10.4. Giao diện cập nhật sản phẩm được quản lý bởi quản trị viên



The screenshot shows the Galito y libros Bookstore dashboard. On the left, there's a sidebar with a menu: three horizontal lines (Dashboard), a search bar, and a 'Orders' option. The main area is titled 'ALL ORDERS' and displays a table of orders:

Order ID	Status	Items Qty	Amount	Action
6199dff25434a22e53df318f	Delivered	1	95,200.384	
6199e3a95434a22e53df323e	Shipped	2	111,891.584	
619a304765329839e7ce5977	Processing	2	201,124.992	

Hình 4.10.5 Giao diện danh sách đơn đặt hàng trên hệ thống được quản lý bởi quản trị viên

Shipping Info

Name: NgQuangVu
Phone: 908556677
Address: ABC, TayNinh, 37, 840000, VN

Payment

PAID
Amount: 201124.992 VND

Order Status

Processing

Your Cart Items:

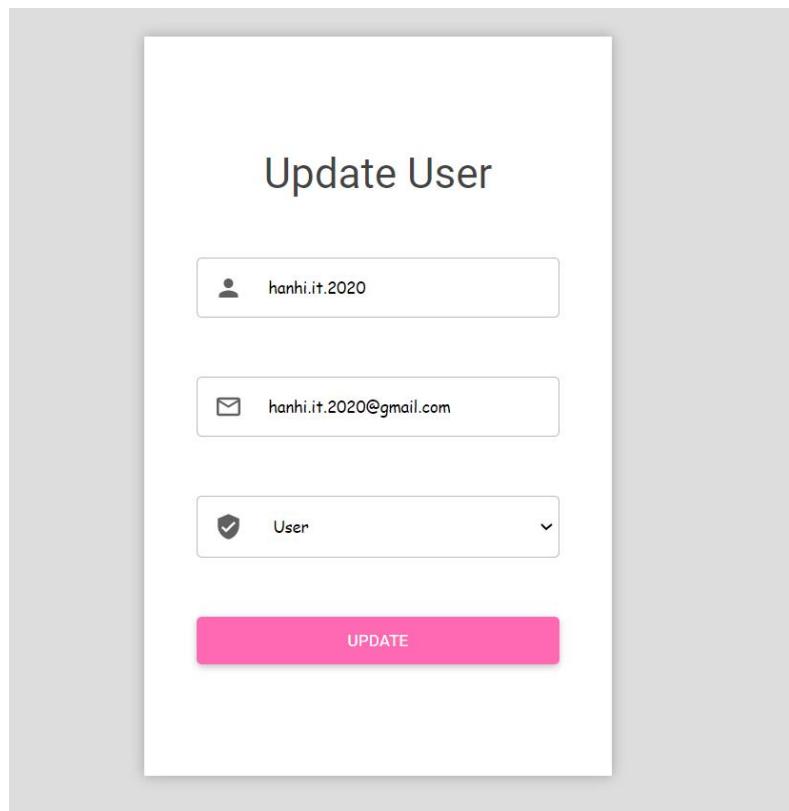
	The C Programming Language. 2nd Edition - Brian Kernighan	2 x 46056 VND = 92112 VND
	Agile Software Development: Principles, Patterns, and Practices - Robert C. Martin	1 x 55471 VND = 55471 VND

Hình 4.10.6 . Giao diện chỉnh sửa trạng thái đơn đặt hàng được quản lý bởi quản trị viên

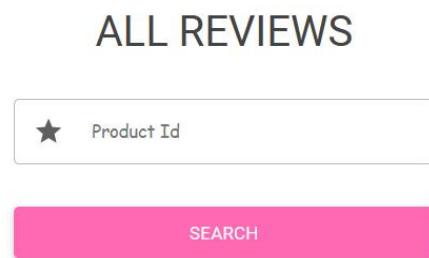
ALL USERS

User ID	Email	Name	Role	Action
618fd529339ac90d6540410c	littlejaycece@gmail.com	littlejaycece	admin	
6194dd0fb4ed77adfd78ce1c	hanhi.it.2020@gmail.com	hanhi.it.2020	user	
619668e1760cc67d62b5eb7a	qvu006@gmail.com	NgQuangVu	user	
6199d6315434a22e53df2fde	18110241@student.hcmute.edu.vn	Quang Vu	user	
6199e90e5434a22e53df33a8	it.me159@gmail.com	Nguyen Vu	admin	
619a8e43cc3a82c1fb556e6	18110168@student.hcmute.edu.vn	incognito user	user	
619ced4ed3da46c387b18462	souldippy@gmail.com	Phuong Chi	user	

Hình 4.10.7. Giao diện danh sách users trên hệ thống được quản lý bởi admin



Hình 4.10.8. Giao diện nâng cấp quyền hạn tài khoản user được quản lý bởi quản trị viên



No Reviews Found

Hình 4.10.9 Giao diện các đánh giá sản phẩm bởi users được quản lý bởi quản trị viên

ALL REVIEWS				
Review ID	User	Comment	Rating	Action
6196aa02a0220de960c290...	NgQuangVu	Máy Xin	5	
< > 1-1 of 1 < >				

Hình 4.10.10. Giao diện danh sách các đánh giá 1 sản phẩm bởi users được quản lý bởi quản trị viên

CHƯƠNG V: ĐỒ THỊ DÒNG ĐIỀU KHIỂN

1. User - Register

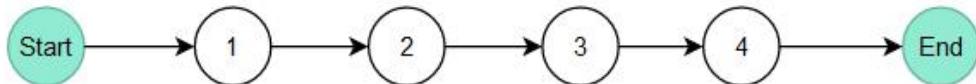
```
//Register a User
exports.registerUser = catchAsyncError(async (req, res, next) => {
  const myCloud = await cloudinary.v2.uploader.upload(req.body.avatar, {
    folder: "avatars",
    width: 150,
    crop: "scale",
  }); /*(1)*/

  const { name, email, password } = req.body; /*(2)*/

  const user = await User.create({
    name,
    email,
    password,
    avatar: {
      public_id: myCloud.public_id,
      url: myCloud.secure_url,
    },
 }); /*(3)*/

  sendToken(user, 201, res); /*(4)*/
});
```

Table 5.1: Code Backend Đăng ký tài khoản



Hình 5.1: Đồ thị dòng điều khiển Đăng ký tài khoản

- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:
 - **1→2→3→4**

2. User - Login

```
//Login User
exports.loginUser = catchAsyncError(async (req, res, next) => {
  const { email, password } = req.body; /*(1)*/

  if (!email /*(2)*/ || !password /*(3)*/) {
    return next(new ErrorHander("Please enter email and password", 400)); /*(4)*/
});
```

```

}

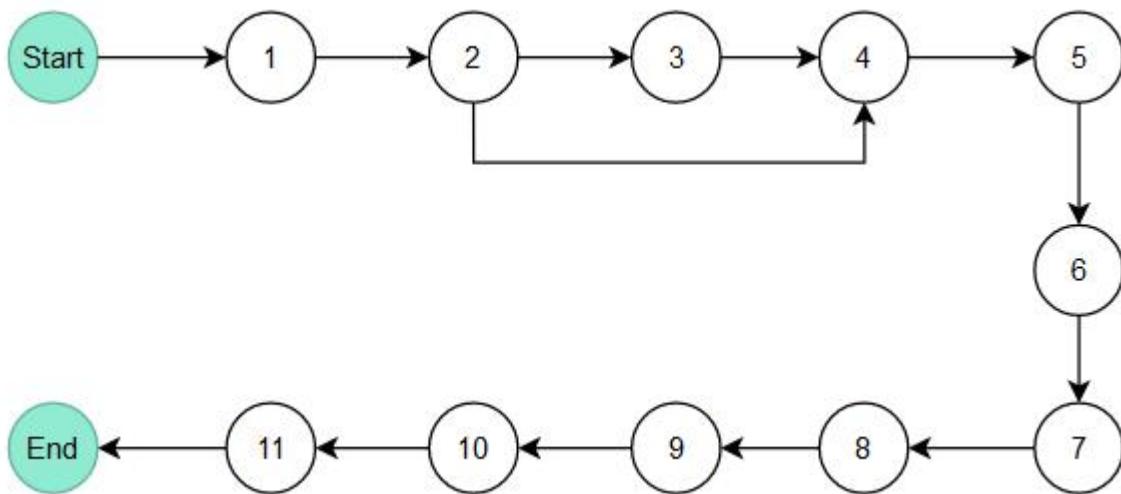
const user = await User.findOne({ email }).select("+password"); /*(5)*/
if (!user /*(6)*/) {
    return next(new ErrorHander("Invalid email or password", 401)); /*(7)*/
}

const isPasswordMatched = await user.comparePassword(password); /*(8)*/
if (!isPasswordMatched /*(9)*/) {
    return next(new ErrorHander("Invalid email or password", 401)); /*(10)*/
}

sendToken(user, 200, res); /*(11)*/
});

```

Table 5.2: Code Backend Đăng nhập tài khoản



Hình 5.2: Đồ thị dòng điều khiển Đăng nhập tài khoản

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:
 - $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11$
 - $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11$

3. User - Forgot Password

```

//Forgot Password
exports.forgotPassword = catchAsyncError(async (req, res, next) => {
    const user = await User.findOne({ email: req.body.email }); /*(1)*/
    if (!user /*(2)*/) {
        return next(new ErrorHander("User not found", 404)); /*(3)*/
    }
}

```

```

const resetToken = user.getResetPasswordToken(); /*(4)*/
await user.save({ validateBeforeSave: false }); /*(5)*/

const resetPasswordUrl = `${req.protocol}://${req.get(
  "host"
)}/api/v1/password/reset/${resetToken}`; /*(6)*/

const message = `Your password reset token is temporary: - \n\n
${resetPasswordUrl} \n\nIf you have not requested this email then, please
ignore it.`; /*(7)*/

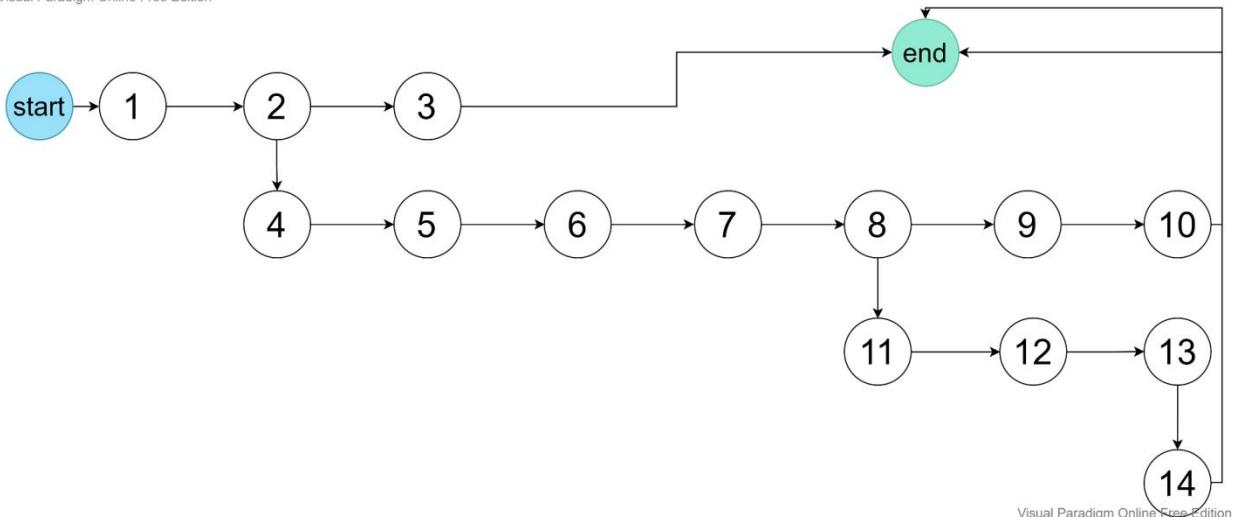
try /*(8)*/ {
  await sendEmail({
    email: user.email,
    subject: 'Website Password Recovery', /*(9)*/,
    message,
  });
}

res.status(200).json({
  success: true, /*(10)*/,
  message: `Email sent to ${user.email} successfully`,
});
} catch (error) {
  user.resetPasswordToken = undefined; /*(11)*/
  user.resetPassordExpire = undefined; /*(12)*/
  await user.save({ validateBeforeSave: false }); /*(13)*/
  return next(new ErrorHander(error.message, 500)); /*(14)*/
}
}
);

```

Table 5.3: Code Backend Quên mật khẩu

Visual Paradigm Online Free Edition



Hình 5.3: Đồ thị dòng điều khiển Quên mật khẩu

- Đồ thị có 2 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 2 = 3$
- Đường đi tuyến tính độc lập là:

1→2→3

1→2→4→5→6→7→8→9→10

1→2→4→5→6→7→8→11→12→13→14

4. User - Reset Password

```
//Reset Password
exports.resetPassword = catchAsyncError(async (req, res, next) => {
  const resetPasswordToken = crypto
    .createHash("sha256")
    .update(req.params.token)
    .digest("hex"); /*(1)*/

  const user = await User.findOne({
    resetPasswordToken,
    resetPasswordExpire: { $gt: Date.now() },
  }); /*(2)*/

  if (!user /*(3)*/) {
    return next(
      new ErrorHander(
        "Reset Password Token is invalid or has been expired",
        400
      )
    ); /*(4)*/
  }

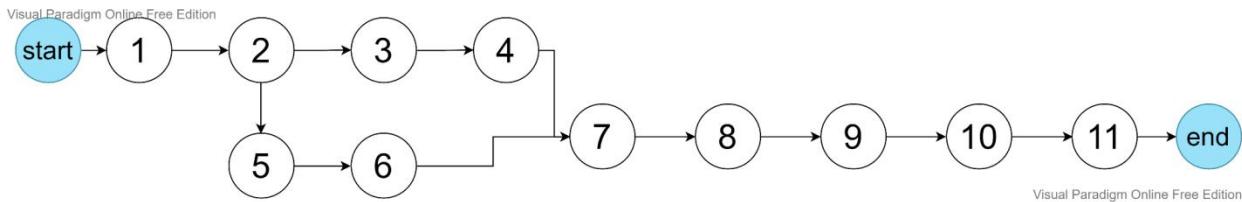
  if (req.body.password !== req.body.confirmPassword /*(5)*/) {
    return next(new ErrorHander("Password does not match", 400)); /*(6)*/
  }

  user.password = req.body.password; /*(7)*/
  user.resetPasswordToken = undefined; /*(8)*/
  user.resetPasswordExpire = undefined; /*(9)*/

  await user.save(); /*(10)*/

  sendToken(user, 200, res); /*(11)*/
});
```

Table 5.4: Code Backend Cài lại mật khẩu



Hình 5.4: Đồ thị dòng điều khiển Cài lại mật khẩu

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:

1→2→3→4→7→8→9→10→11

1→2→5→6→7→8→9→10→11

5. User - See All Products

```
//Get All Products
exports.getAllProducts = catchAsyncError(async (req, res, next) => {
  const resultPerPage = 8; /*(1)*/
  const productsCount = await Product.countDocuments(); /*(2)*/

  const apiFeature = new ApiFeatures(Product.find(), req.query)
    .search()
    .filter(); /*(3)*/

  let products = await apiFeature.query; /*(4)*/

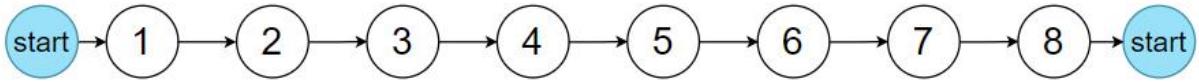
  let filteredProductsCount = products.length; /*(5)*/

  apiFeature.pagination(resultPerPage); /*(6)*/

  products = await apiFeature.query.clone(); /*(7)*/

  res.status(200).json({
    success: true,
    products,
    productsCount,
    resultPerPage,
    filteredProductsCount,
  }); /*(8)*/
});
```

Table 5.5: Code Backend Xem tất cả các sản phẩm



Hình 5.5: Đồ thị dòng điều khiển Xem tất cả sản phẩm

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:

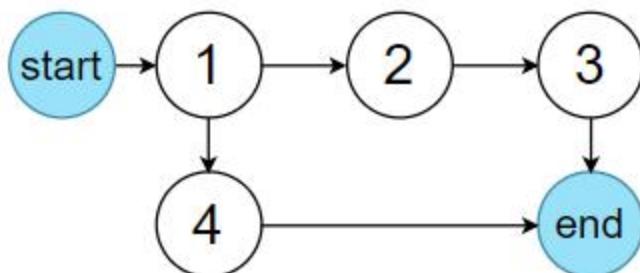
1→2→3→4→5→6→7→8

6. User - See A Product Details

```

//Get A Product Details
exports.getProductDetails = catchAsyncError(async (req, res, next) => {
  const product = await Product.findById(req.params.id); /*(1)*/
  if (!product /*(2)*{
    return next(new ErrorHander("Product not found", 404)); /*(3)*
  }
  res.status(200).json({
    success: true,
    product,
  }); /*(4)*/
}); /*(4)*/
```

Table 5.6: Code Backend Xem thông tin chi tiết 1 sản phẩm



Hình 5.6: Đồ thị dòng điều khiển Xem thông tin chi tiết 1 sản phẩm

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:

1→2→3

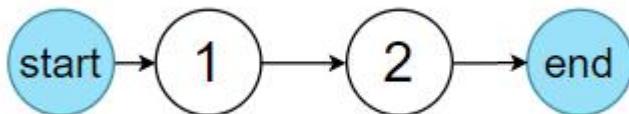
1→4

10. Customer - Log out

```
//Logout User
exports.logout = catchAsyncError(async (req, res, next) => {
  res.cookie("token", null, {
    expires: new Date(Date.now()),
    httpOnly: true,
  });
}); /*(1)*/

res.status(200).json({
  success: true,
  message: "Logout successfully",
}); /*(2)*/
});
```

Table 5.10: Code Backend Đăng xuất tài khoản



Hình 5.10: Đồ thị dòng điều khiển Đăng xuất tài khoản

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 0 = 1$
- Đường đi tuyến tính độc lập là:

1→2

11. Customer - See Profile Details

```
// Get User Detail
exports.getUserDetails = catchAsyncError(async (req, res, next) => {
  const user = await User.findById(req.user.id); /*(1)*/

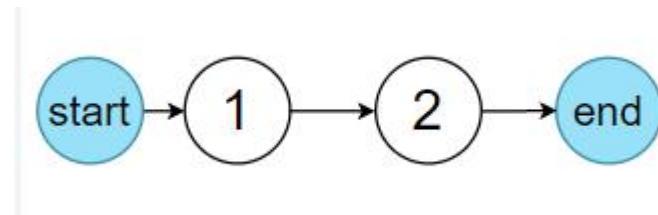
  res.status(200).json({
```

```

        success: true,
        user,
    });
}); /*(2)*/
});

```

Table 5.11: Code Backend Xem thông tin chi tiết tài khoản



Hình 5.11: Đồ thị dòng điều khiển Xem thông tin chi tiết tài khoản

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 0 = 1$
- Đường đi tuyến tính độc lập là:

1→2

12. Customer - Update Profile

```

// Update User Profile
exports.updateProfile = catchAsyncError(async (req, res, next) => {
    const newUserData = {
        name: req.body.name,
        email: req.body.email,
    }; /*(1)*/

    //cloudinary:
    if (req.body.avatar !== "") /*(2)*/ {
        const user = await User.findById(req.user.id); /*(3)*/

        const imageld = user.avatar.public_id; /*(4)*/ //lấy avatar hiện tại
        await cloudinary.v2.uploader.destroy(imageld); /*(5)*/ //xóa avatar hiện tại

        const myCloud = await cloudinary.v2.uploader.upload(req.body.avatar, {
            folder: "avatars",
            width: 150,
            crop: "scale",
       }); /*(6)*/ //tải avatar mới lên cloudinary
    }
});

```

```

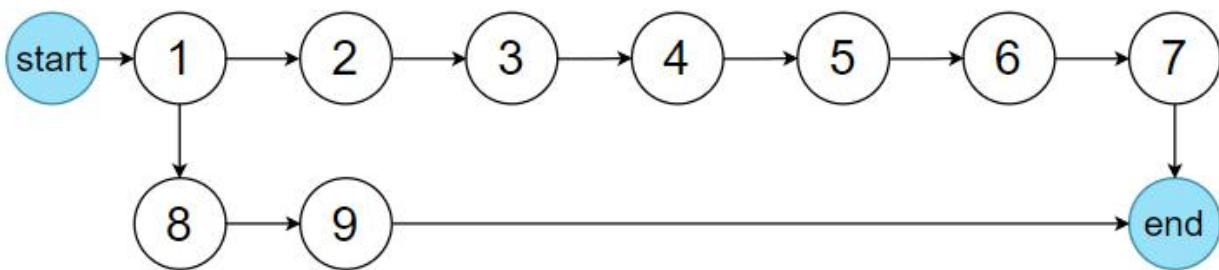
newUserData.avatar = {
  public_id: myCloud.public_id,
  url: myCloud.secure_url,
}; /*(7)*/
}

const user = await User.findByIdAndUpdate(req.user.id, newUserData, {
  new: true,
  runValidators: true,
  useFindAndModify: false,
}); /*(8)*/

res.status(200).json({
  success: true,
}); /*(9)*/
);

```

Table 5.12: Code Backend Cập nhật thông tin tài khoản



Hình 5.12: Đồ thị dòng điều khiển Cập nhật thông tin tài khoản

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:

1→2→3→4→5→6→7

1→8→9

13. Customer - Update Password

```

// Update User password
exports.updatePassword = catchAsyncError(async (req, res, next) => {
  const user = await User.findById(req.user.id).select("+password"); /*(1)*/

  const isPasswordMatched = await user.comparePassword(req.body.oldPassword); /*(2)*/

  if (!isPasswordMatched /*(3)*/) {
    return next(new ErrorHander("Old password is incorrect", 400)); /*(4)*/
  }
}

```

```

if (req.body.newPassword !== req.body.confirmPassword) {  

    return next(new ErrorHandler("password does not match", 400)); /*(6)*/  

}  
  

user.password = req.body.newPassword; /*(7)*/  
  

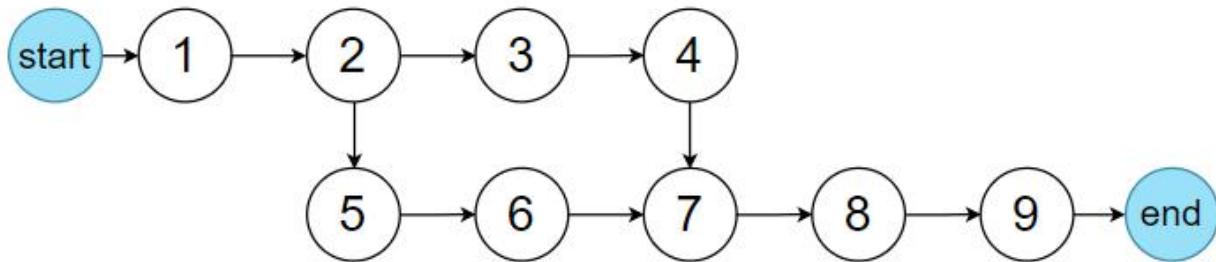
await user.save(); /*(8)*/  
  

sendToken(user, 200, res); /*(9)*/  

);

```

Table 5.13: Code Backend Cập nhật mật khẩu tài khoản



Hình 5.13: Đồ thị dòng điều khiển Cập nhật mật khẩu tài khoản

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyền tính độc lập là:

1→2→3→4→7→8→9

1→2→5→6→7→8→9

14. Customer - Create A Product Review

```

//Create A New Review or Update A Review
exports.createProductReview = catchAsyncError(async (req, res, next) => {
  const { rating, comment, productId } = req.body; /*(1)*/  
  

  const review = {  

    user: req.user._id,  

    name: req.user.name,  

    rating: Number(rating),  

    comment,  

  }; /*(2)*/  
  

  const product = await Product.findById(productId); /*(3)*/

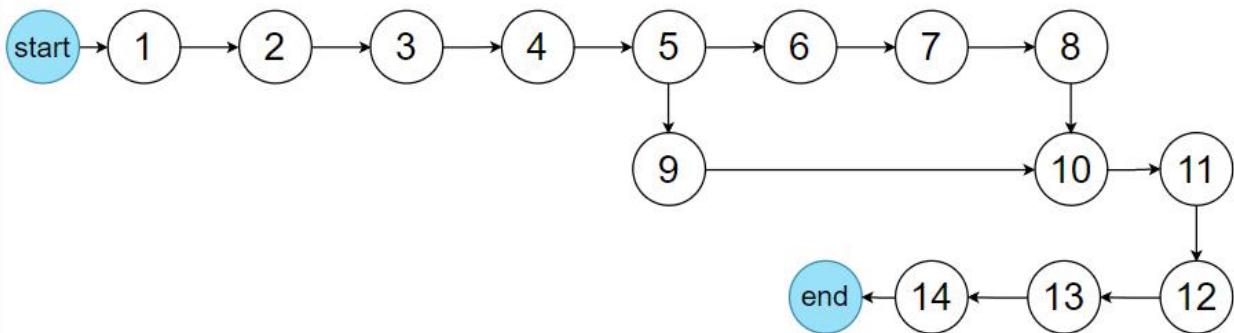
```

```

const isReviewed = product.reviews.find(
  (rev) => rev.user.toString() === req.user._id.toString()
); /*(4)*/
if (isReviewed /*(5)*/) {
  product.reviews.forEach((rev) => {
    if (rev.user.toString() === req.user._id.toString()) /*(6)*/
      (rev.rating = rating /*(7)*), (rev.comment = comment /*(8)*);
  });
} else {
  product.reviews.push(review);
  product.numOfReviews = product.reviews.length; /*(9)*/
}
let avg = 0; /*(10)*/
product.reviews.forEach((rev) => {
  avg += rev.rating;
}); /*(11)*/
product.ratings = avg / product.reviews.length; /*(12)*/
await product.save({ validateBeforeSave: false }); /*(13)*/
res.status(200).json({
  success: true,
}); /*(14)*/
});

```

Table 5.14: Code Backend Đánh giá 1 sản phẩm



Hình 5.14: Đồ thị dòng điều khiển Đánh giá 1 sản phẩm

- Đồ thị có 1 nút quyết định nhị phân.

- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:

1→2→3→4→7→8→10→11→12→13→14

1→2→3→5→9→10→11→12→13→14

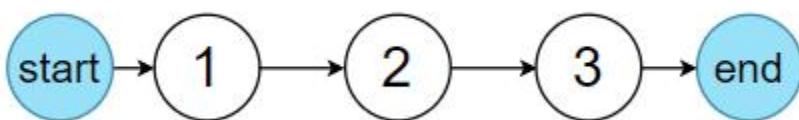
16. Customer - Take an order

```
// Create A New Order
exports.newOrder = catchAsyncError(async (req, res, next) => {
  const {
    shippingInfo,
    orderItems,
    paymentInfo,
    itemsPrice,
    taxPrice,
    shippingPrice,
    totalPrice,
  } = req.body; /*(1)*/

  const order = await Order.create({
    shippingInfo,
    orderItems,
    paymentInfo,
    itemsPrice,
    taxPrice,
    shippingPrice,
    totalPrice,
    paidAt: Date.now(),
    user: req.user._id,
  }); /*(2)*/

  res.status(201).json({
    success: true,
    order,
 }); /*(3)*/
});
```

Table 5.16: Code Backend Đặt đơn hàng



Hình 5.16: Đồ thị dòng điều khiển Đặt đơn hàng

- Đồ thị có 1 nút quyết định nhị phân.

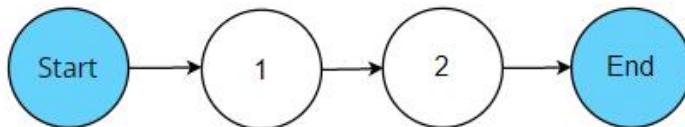
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:

1→2→3

17. Customer - Check out

```
exports.processPayment = catchAsyncError(async (req, res, next) => {
  const myPayment = await stripe.paymentIntents.create({
    amount: req.body.amount,
    currency: "vnd",
    metadata: {
      company: "Gatitos Group",
    },
  });
  /*(1)*/
  res
    .status(200)
    .json({ success: true, client_secret: myPayment.client_secret });
  /*(2)*/
});
```

Table 5.17: Code Backend Thanh toán đơn hàng



Hình 5.17: Đồ thị dòng điều khiển Thanh toán đơn hàng

- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là: $1 \rightarrow 2$

18. Customer - See all orders

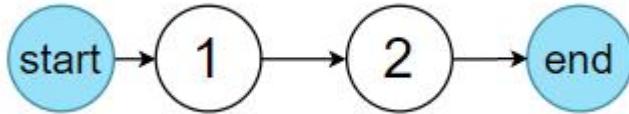
```
//Get logged in user Orders
exports.myOrders = catchAsyncError(async (req, res, next) => {
  const orders = await Order.find({ user: req.user._id });
  /*(1)*/
});
```

```

res.status(200).json({
    success: true,
    orders,
}); /*(2)*/
});

```

Table 5.18: Code Backend Xem tất cả các đơn hàng



Hình 5.18: Đồ thị dòng điều khiển Xem tất cả các đơn hàng

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 0 = 1$
- Đường đi tuyến tính độc lập là:

1→2

19. Customer - See a single order

```

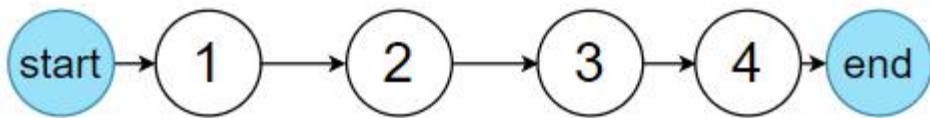
//Get A Single Order
exports.getSingleOrder = catchAsyncError(async (req, res, next) => {
    const order = await Order.findById(req.params.id).populate(
        "user",
        "name email"
); /*(1)*/

    if (!order /*(2)*/) {
        return next(new ErrorHander("Order not found with this Id", 404)); /*(3)*/
    }

    res.status(200).json({
        success: true,
        order,
   }); /*(4)*/
});

```

Table 5.19: Code Backend Xem thông tin 1 đơn hàng



Hình 5.19: Đồ thị dòng điều khiển Xem thông tin 1 đơn hàng

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 0 = 1$
- Đường đi tuyến tính độc lập là:

1→2→3→4

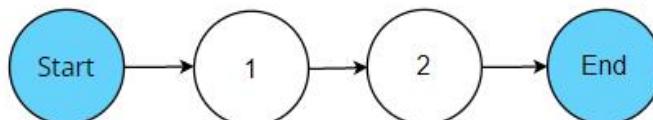
20. Admin - See All Users

```

// Get all users (admin)
exports.getAllUser = catchAsyncError(async (req, res, next) => {
  const users = await User.find(); /*(1)*/
  res.status(200).json({
    success: true,
    users,
  }); /*(2)*/
});

```

Table 5.20: Code Backend Xem tất cả các người dùng



Hình 5.21: Đồ thị dòng điều khiển Xem tất cả các người dùng

- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là: $1 \rightarrow 2$

21. Admin - Update User Role

```
// Update User Role (admin)
exports.updateUserRole = catchAsyncError(async (req, res, next) => {
  const newData = {
    name: req.body.name,
    email: req.body.email,
    role: req.body.role,
  }; /*(1)*/

  await User.findByIdAndUpdate(req.params.id, newData, {
    new: true,
    runValidators: true,
    useFindAndModify: false,
}); /*(2)*/

  res.status(200).json({
    success: true,
  }); /*(3)*/
});
```

Table 5.21: Code Backend Nâng cấp quyền hạn người dùng



Hình 5.21: Đồ thị dòng điều khiển Nâng cấp quyền hạn người dùng

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:

1→2→3

22. Admin - Delete Users

```
// Delete User (admin)
exports.deleteUser = catchAsyncError(async (req, res, next) => {
  const user = await User.findById(req.params.id); /*(1)*/

  if (!user /*(2)*/) {
    return next(
      new ErrorHander(`User does not exist with Id: ${req.params.id}`, 400)
    ); /*(3)*/
}
```

```

const imageUrl = user.avatar.public_id; /*(4)*/

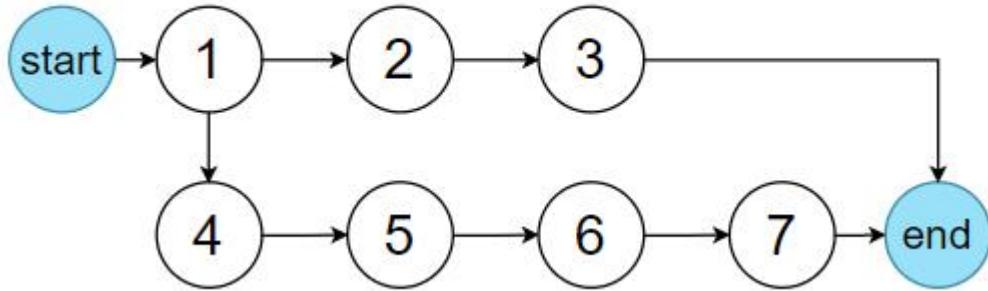
await cloudinary.v2.uploader.destroy(imageId); /*(5)*/

await user.remove(); /*(6)*/

res.status(200).json({
  success: true,
  message: "User deleted successfully",
}); /*(7)*/
});

```

Table 5.22: Code Backend Xóa người dùng



Hình 5.21: Đồ thị dòng điều khiển Nâng cấp quyền hạn người dùng

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:

1→2→3

1→4→5→6→7

23. Admin - See All Products

```

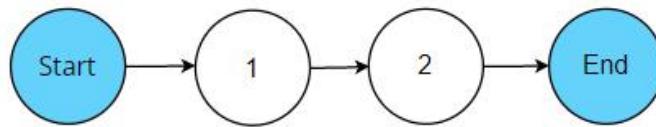
//Get All Product - Admin Role
exports.getAdminProducts = catchAsyncError(async (req, res, next) => {
  const products = await Product.find(); /*(1)*/

  res.status(200).json({
    success: true,
    products,
  });
});

```

```
}); /*(2)*/
});
```

Table 5.23: Code Backend Xem tất cả các sản phẩm



Hình 5.23: Đồ thị dòng điều khiển Xem tất cả các sản phẩm

- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là: $1 \rightarrow 2$

24. Admin - Create Products

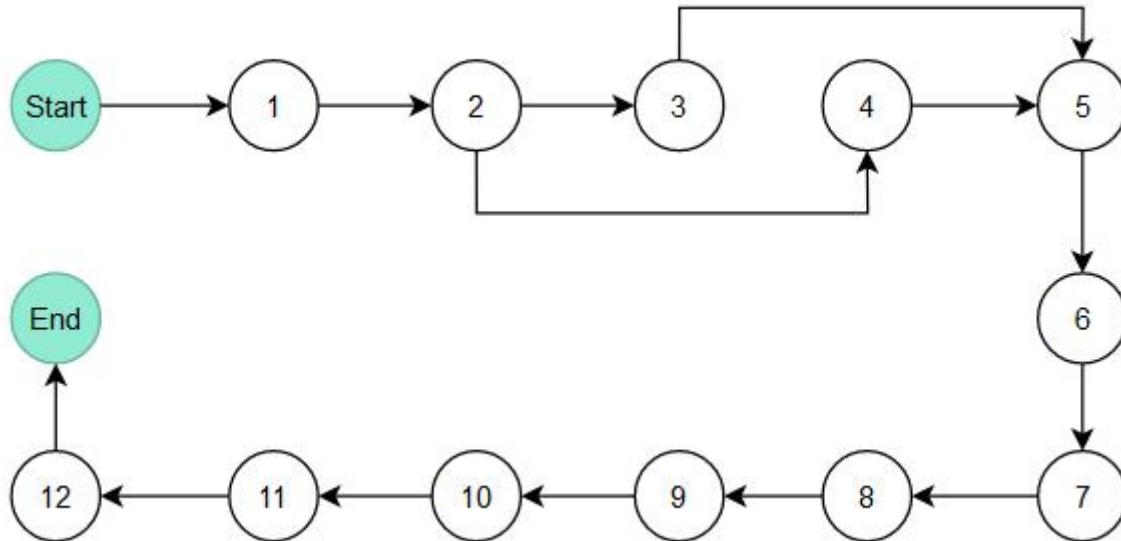
```
//Create A Product - Admin Role
exports.createProduct = catchAsyncError(async (req, res, next) => {
  let images = []; /*(1)*/
  if (typeof req.body.images === "string" /*(2)*{
    images.push(req.body.images); /*(3)*/
  } else {
    images = req.body.images; /*(4)*/
  }
  const imagesLinks = []; /*(5)*/
  for (let i = 0; i < images.length; i++) /*(6)*{
    const result = await cloudinary.v2.uploader.upload(images[i], {
      folder: "products",
    }); /*(7)*/
    imagesLinks.push({
      public_id: result.public_id,
      url: result.secure_url,
    }); /*(8)*/
  }
})
```

```

req.body.images = imagesLinks; /*(9)*/
req.body.user = req.user.id; /*(10)*/
const product = await Product.create(req.body); /*(11)*/
res.status(201).json({
  success: true,
  product,
}); /*(12)*/
});

```

Table 5.24: Code Backend Tạo sản phẩm mới



Hình 5.24: Đồ thị dòng điều khiển Tạo sản phẩm mới

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:
 - **1 → 2 → 3 → 5 → 6 → 8 → 9 → 10 → 11 → 12**
 - **1 → 2 → 4 → 5 → 6 → 8 → 9 → 10 → 11 → 12**

25. Admin - Update Products

```

//Update A Product - Admin Role
exports.updateProduct = catchAsyncError(async (req, res, next) => {
  let product = await Product.findById(req.params.id); /*(1)*/
  if (!product /*(2)*{
    return next(new ErrorHander("Product not found", 404)); /*(3)*/
  }

```

```

}

// Images Update:
let images = []; /*(4)*/

if (typeof req.body.images === "string" /*(5)*/) {
    images.push(req.body.images); /*(6)*/
} else {
    images = req.body.images; /*(7)*/
}

if (images !== undefined /*(8)*/) {
    // Deleting Images From Cloudinary
    for (let i = 0; i < product.images.length; i++ /*(9)*/) {
        await cloudinary.v2.uploader.destroy(product.images[i].public_id); /*(10)*/
    }

    const imagesLinks = []; /*(11)*/

    for (let i = 0; i < images.length; i++ /*(12)*/) {
        const result = await cloudinary.v2.uploader.upload(images[i], {
            folder: "products",
       }); /*(13)*/

        imagesLinks.push({
            public_id: result.public_id,
            url: result.secure_url,
       }); /*(14)*/
    }

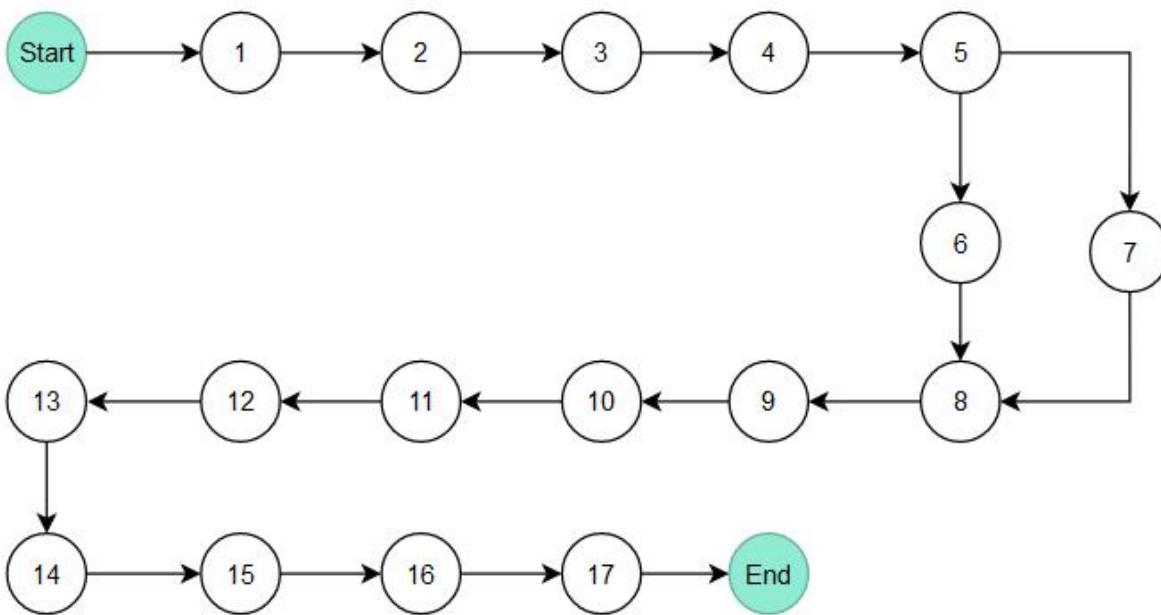
    req.body.images = imagesLinks; /*(15)*/
}

product = await Product.findByIdAndUpdate(req.params.id, req.body, {
    new: true,
    runValidators: true,
    useFindAndModify: false,
}); /*(16)*/

res.status(200).json({
    success: true,
    product,
}); /*(17)*/
}

```

Table 5.25: Code Backend Cập nhật 1 sản phẩm



Hình 5.25: Đồ thị dòng điều khiển Cập nhật 1 sản phẩm

- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:
 - $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 13 \rightarrow 14 \rightarrow 15 \rightarrow 16 \rightarrow 17$
 - $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 13 \rightarrow 14 \rightarrow 15 \rightarrow 16 \rightarrow 17$

26. Admin - Delete Products

```
//Delete A Product - Admin Role
exports.deleteProduct = catchAsyncError(async (req, res, next) => {
  const product = await Product.findById(req.params.id); /*(1)*/
  if (!product /*(2)*{
    return res.status(500).json({
      success: false,
      message: "Product not found",
   }); /*(3)*/
  }

  // Deleting Images From Cloudinary
  for (let i = 0; i < product.images.length; i++ /*(4)*{
    await cloudinary.v2.uploader.destroy(product.images[i].public_id); /*(5)*/
  }

  await product.remove(); /*(6)*

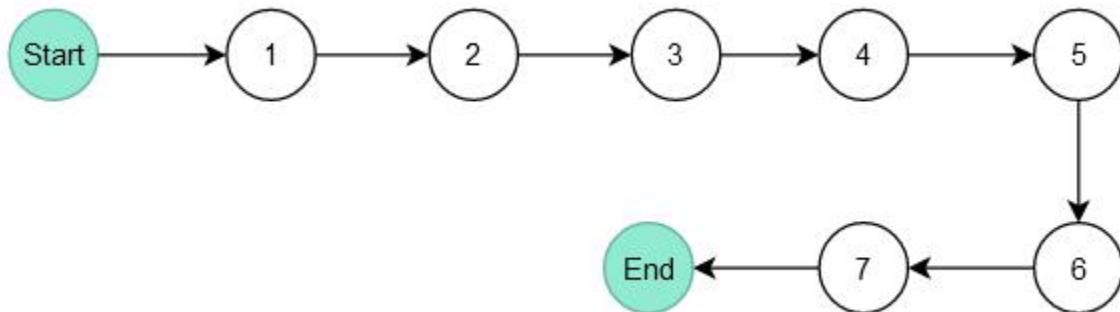
  res.status(200).json({
    success: true,
    message: "Product deleted successfully"
  })
})
```

```

        success: true,
        message: "Product Delete Successfully",
    });
}); /*(7)*/
});
}

```

Table 5.26: Code Backend Xóa sản phẩm



Hình 5.26: Đồ thị dòng điều khiển Xóa sản phẩm

- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:
 - **1→2→3→4→5→6→7**

27. Admin - See All Orders on system

```

//Get all Orders - Admin Role
exports.getAllOrders = catchAsyncError(async (req, res, next) => {
  const orders = await Order.find(); /*(1)*/

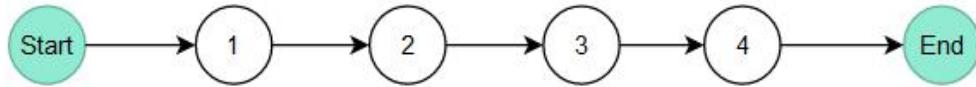
  let totalAmount = 0; /*(2)*/

  orders.forEach((order) => {
    totalAmount += order.totalPrice;
  }); /*(3)*/

  res.status(200).json({
    success: true,
    totalAmount,
    orders,
  }); /*(4)*/
});

```

Table 5.27: Code Backend Xem tất cả các đơn hàng trên hệ thống



Hình 5.27: Đồ thị dòng điều khiển Xem tất cả các đơn hàng trên hệ thống

- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:
 - **1→2→3→4**

28. Admin - Update An Order

```

//Update Order Status - Admin Role
exports.updateOrder = catchAsyncError(async (req, res, next) => {
  const order = await Order.findById(req.params.id); /*(1)*/

  if (!order /*(2)*{
    return next(new ErrorHander("Order not found with this Id", 404)); /*(3)*/

  }

  if (order.orderStatus === "Delivered" /*(4)*{
    return next(new ErrorHander("You have already delivered this order", 400)); /*(5)*/

  }

  if (req.body.status === "Shipped" /*(6)*{
    order.orderItems.forEach(async (o) => {
      await updateStock(o.product, o.quantity);
    }); /*(7)*/

  }
  order.orderStatus = req.body.status; /*(8)*/

  if (req.body.status === "Shipped" /*(9)*{
    order.orderItems.forEach(async (o) => {
      await updateStock(o.product, o.quantity);
   }); /*(10)*/

  }

  if (req.body.status === "Delivered" /*(11)*{
    order.deliveredAt = Date.now(); /*(12)*/

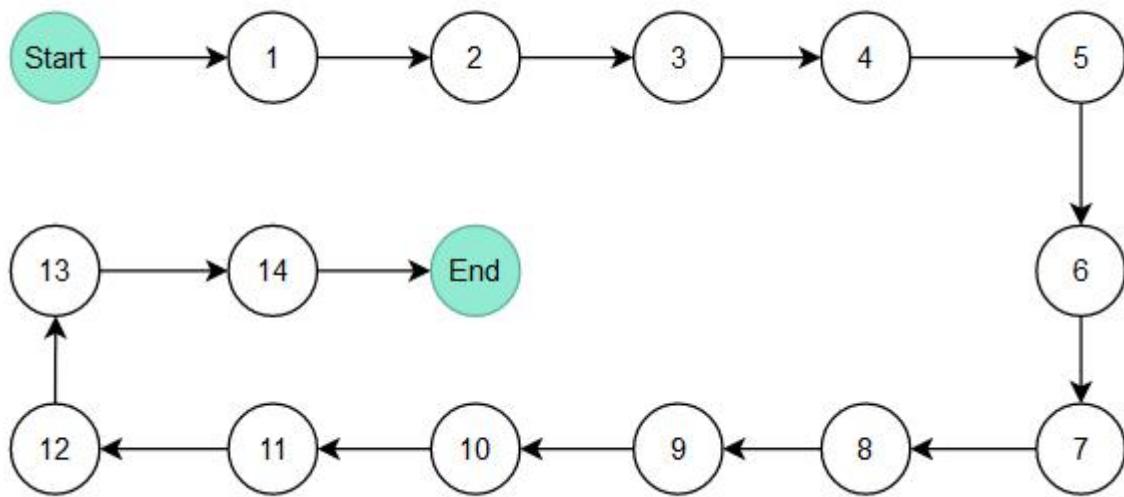
  }

  await order.save({ validateBeforeSave: false }); /*(13)*/
  res.status(200).json({
    success: true,
 }); /*(14)*/

});

```

Table 5.28: Code Backend Cập nhật trạng thái đơn hàng



Hình 5.28: Đồ thị dòng điều khiển Cập nhật trạng thái đơn hàng

- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:
 - **1→2→3→4→5→6→7→8→9→10→11→12→13→14**

29. Admin - Delete An Order

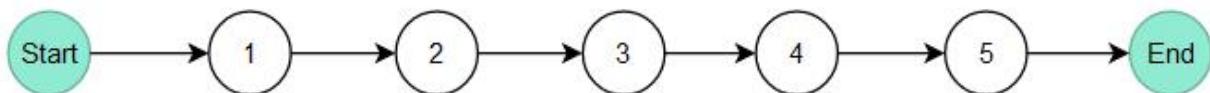
```
//Delete Order - Admin Role
exports.deleteOrder = catchAsyncError(async (req, res, next) => {
  const order = await Order.findById(req.params.id); /*(1)*/

  if (!order /*(2)*/) {
    return next(new ErrorHandler("Order not found with this Id", 404)); /*(3)*/
  }

  await order.remove(); /*(4)*/

  res.status(200).json({
    success: true,
  }); /*(5)*/
});
```

Table 5.29: Code Backend Xóa đơn hàng



Hình 5.29: Đồ thị dòng điều khiển Xóa đơn hàng

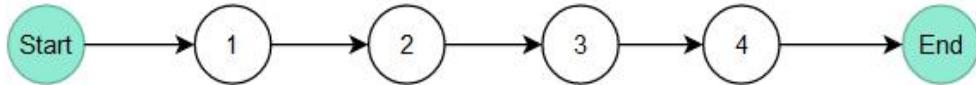
- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:
 - $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$

30. Admin - See All Reviews of a Product

```
//Get All Reviews of A Product
exports.getProductReviews = catchAsyncError(async (req, res, next) => {
  const product = await Product.findById(req.query.id); /*(1)*/
  if (!product /*(2)*{
    return next(new ErrorHander("Product not found", 404)); /*(3)*
  }

  res.status(200).json({
    success: true,
    reviews: product.reviews,
 }); /*(4)*/
});
```

Table 5.30: Code Backend Xem tất cả các đánh giá của 1 sản phẩm



Hình 5.30: Đồ thị dòng điều khiển Xem tất cả các đánh giá của 1 sản phẩm

- Đồ thị có 0 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 0 + 1 = 1$
- Đường đi tuyến tính độc lập là:
 - $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$

31. Admin - Delete A Product Review

```
//Delete A Review
exports.deleteReview = catchAsyncError(async (req, res, next) => {
  const product = await Product.findById(req.query.productId); /*(1)*/
  if (!product /*(2)*{
    return next(new ErrorHander("Product not found", 404)); /*(3)*
  }
```

```

const reviews = product.reviews.filter(
  (rev) => rev._id.toString() !== req.query.id.toString()
); /*(4)*/

let avg = 0; /*(5)*/

reviews.forEach((rev) => {
  avg += rev.rating;
}); /*(6)*/

let ratings = 0; /*(7)*/

if (reviews.length === 0 /*(8)*/) {
  ratings = 0; /*(9)*/
} else {
  ratings = avg / reviews.length; /*(10)*/
}

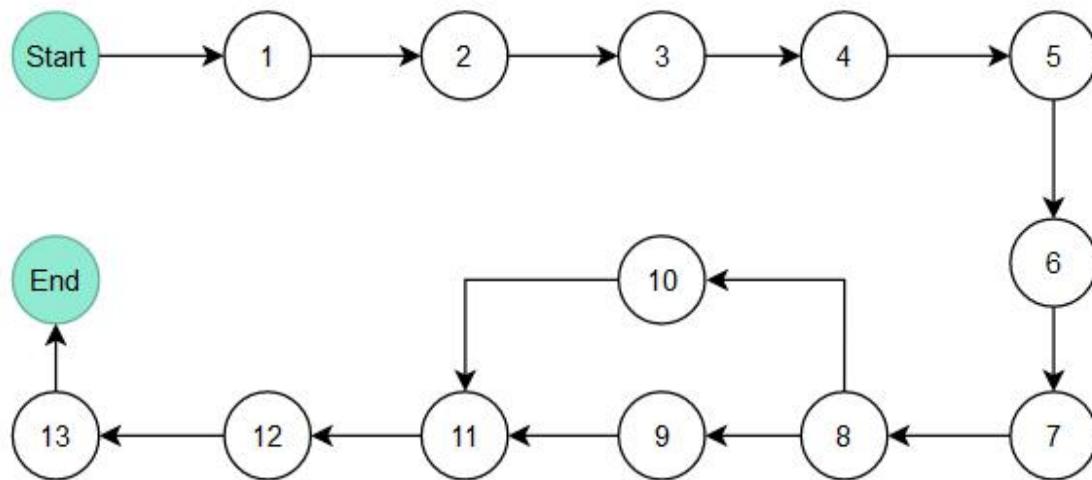
const numOfReviews = reviews.length; /*(11)*/

await Product.findByIdAndUpdate(
  req.query.productId,
  {
    reviews,
    ratings,
    numOfReviews,
  },
  {
    new: true,
    runValidators: true,
    useFindAndModify: false,
  }
); /*(12)*/

res.status(200).json({
  success: true,
}); /*(13)*/
);

```

Table 5.31: Code Backend Xóa đánh giá



Hình 5.31: Đồ thị dòng điều khiển Xóa đánh giá

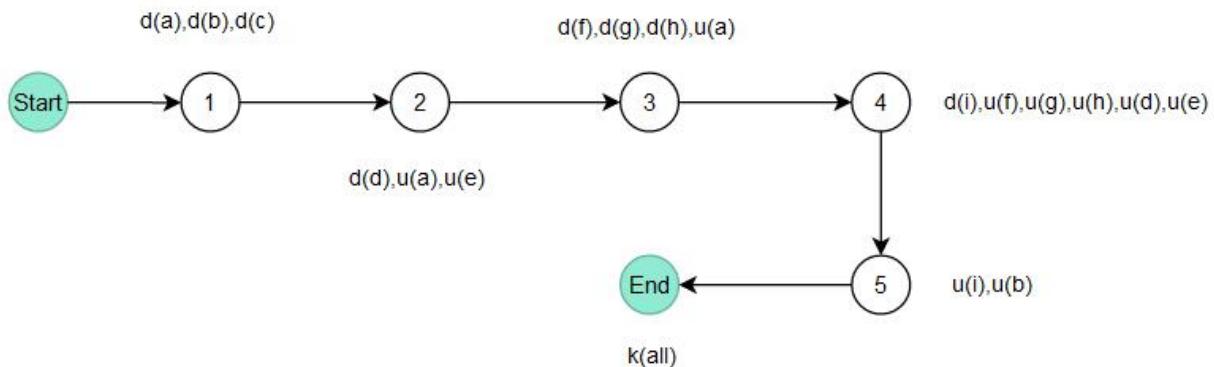
- Đồ thị có 1 nút quyết định nhị phân.
- Độ phức tạp của đồ thị là: $C = 1 + 1 = 2$
- Đường đi tuyến tính độc lập là:
 - **1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 11 → 12 → 13**
 - **1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 10 → 11 → 12 → 13**

CHƯƠNG VI: ĐỒ THỊ DÒNG DỮ LIỆU - KIỂM THỦ ĐỜI SỐNG CÁC BIẾN

1. User - Register

```
//Register a User
exports.registerUser = catchAsyncError(async (req, res, next/*(1)*) => {
  const myCloud = await cloudinary.v2.uploader.upload(req.body.avatar, {
    folder: "avatars",
    width: 150,
    crop: "scale",
 }); /*(2)*/
  const { name, email, password } = req.body; /*(3)*/
  const user = await User.create({
    name,
    email,
    password,
    avatar: {
      public_id: myCloud.public_id,
      url: myCloud.secure_url,
    },
 }); /*(4)*/
  sendToken(user, 201, res); /*(5)*/
});
```

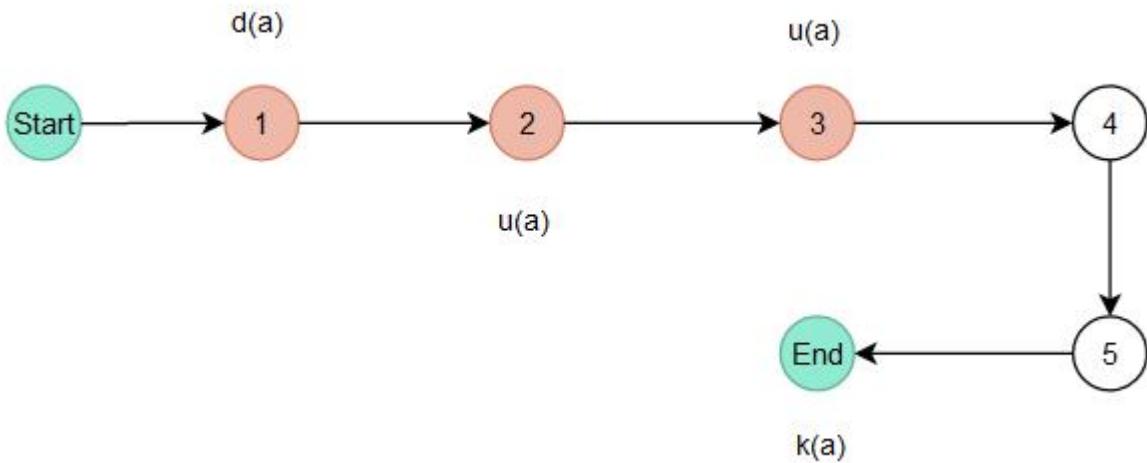
Table 6.1: Code Backend Đăng ký tài khoản



Hình 6.1: Đồ thị dòng dữ liệu Đăng ký tài khoản

Kiểm thử đời sống 9 biến: a(req), b(res), c(next), d(myCloud), e/avatar), f(name), g(email), h(password), i(user)

- Kiểm thử đời sống biển a(req):

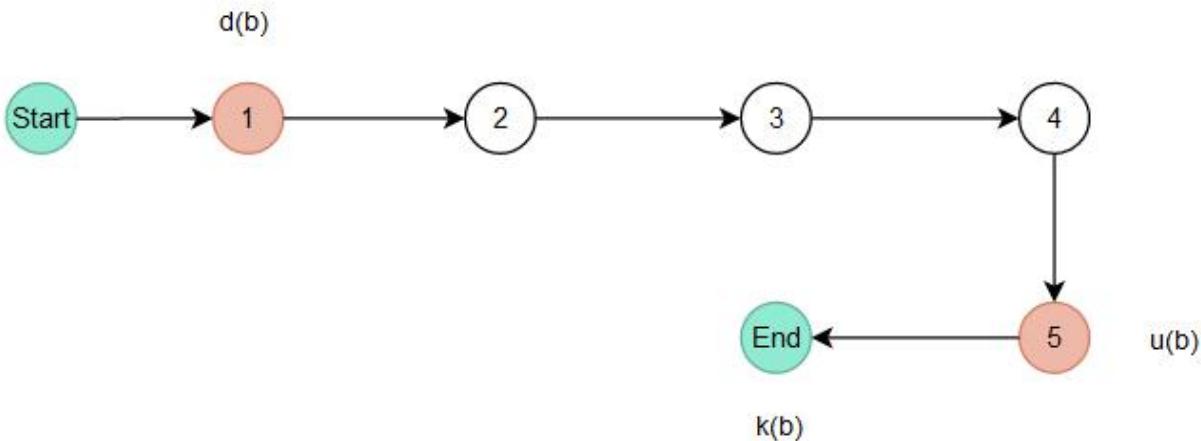


Hình 6.1.a: Đồ thị đời sống biển a(req) Đăng ký tài khoản

Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biển b(res):

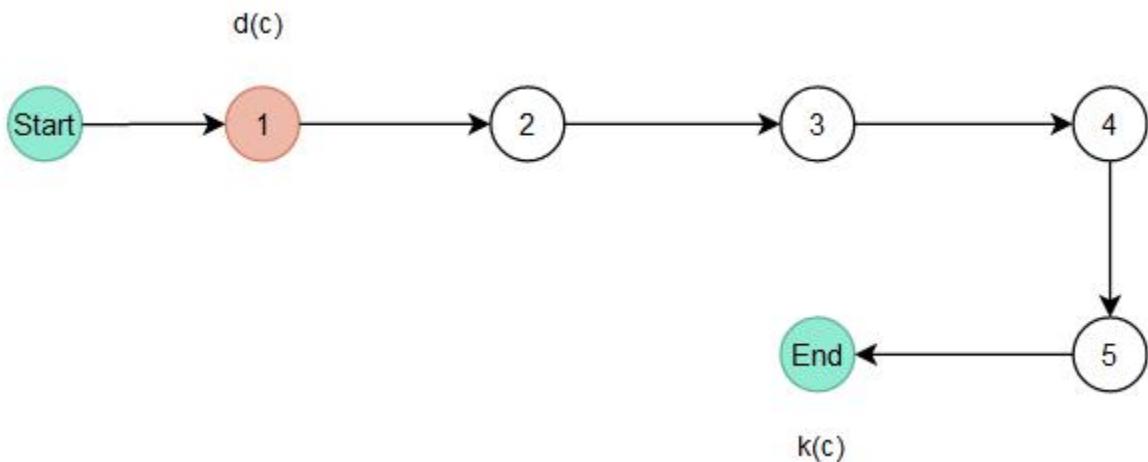


Hình 6.1.b: Đồ thị đời sống biển b(res) Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến $c(next)$:

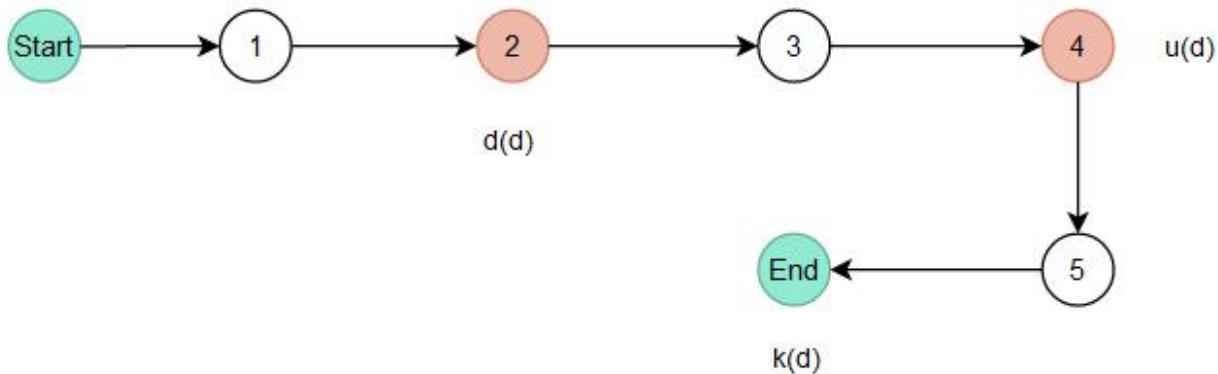


Hình 6.1.c: Đồ thị đời sống biến $c(next)$ Đăng ký tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến $d(myCloud)$:

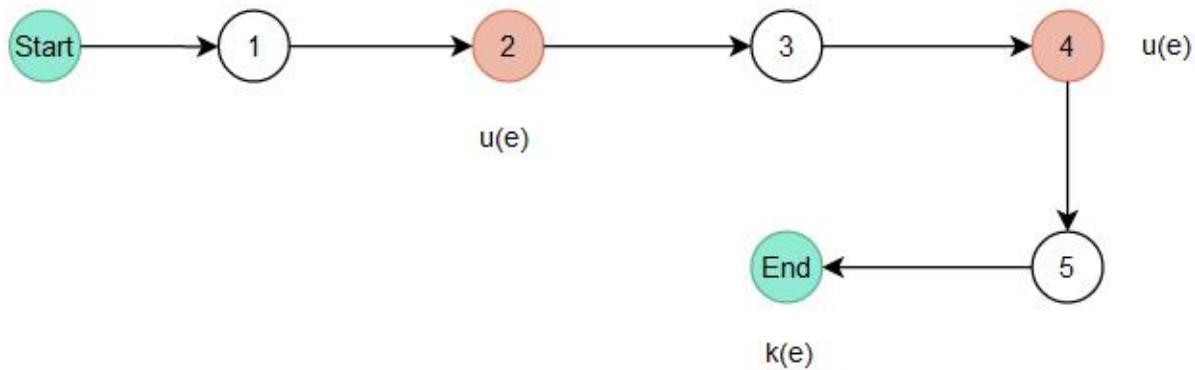


Hình 6.1.d: Đồ thị đời sống biến $d(myCloud)$ Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biển e (avatar):

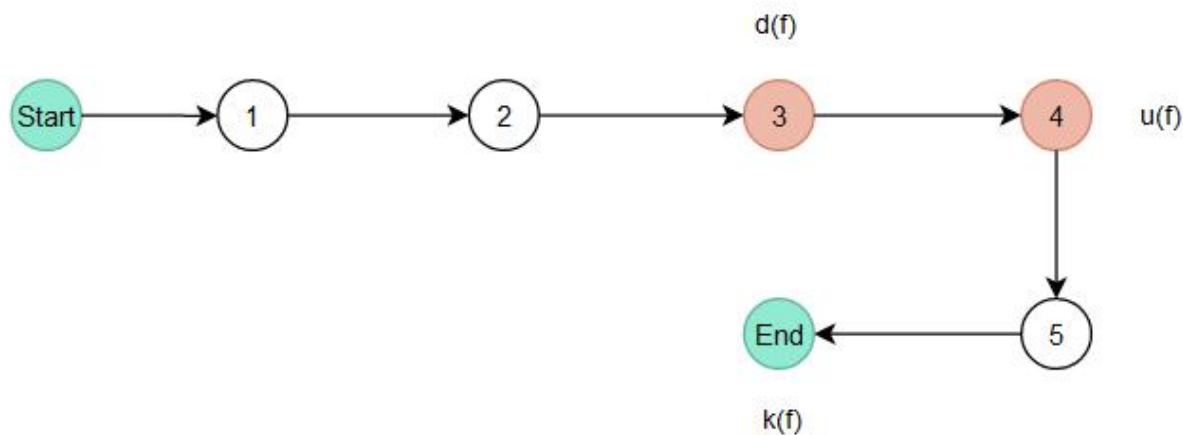


Hình 6.1.e: Đồ thị đời sống biển e (avatar) Đăng ký tài khoản

Kịch bản 1: ~uuuk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biển chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biển $f(name)$:

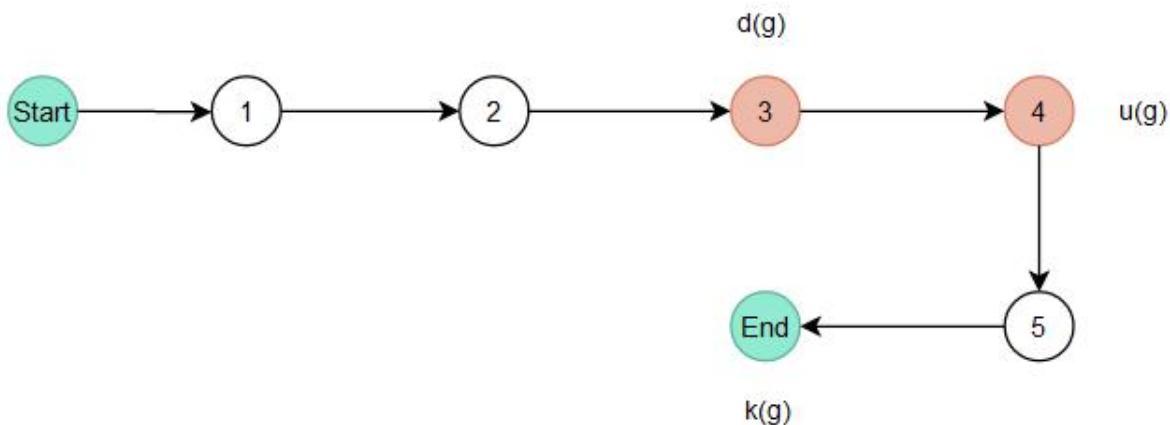


Hình 6.1.f: Đồ thị đời sống biển $f(name)$ Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đổi sống biển g(email):

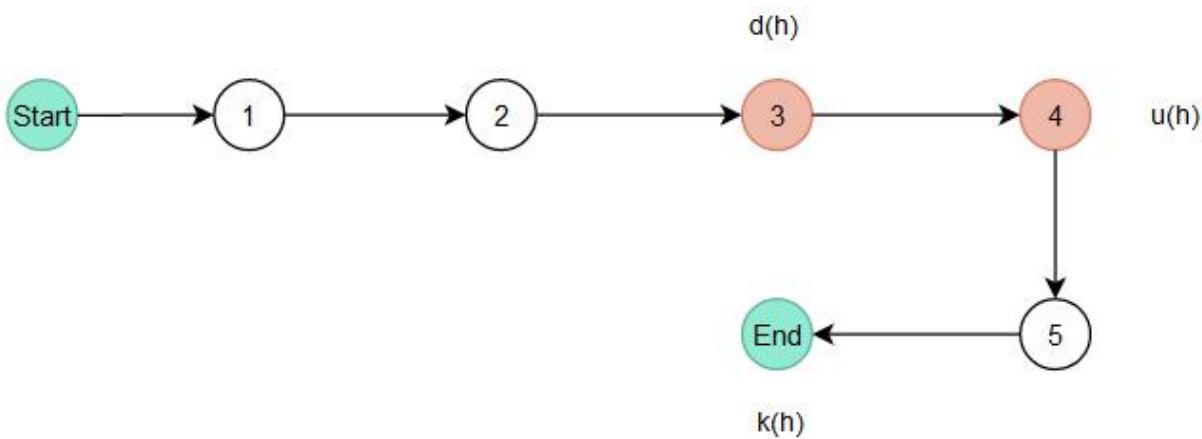


Hình 6.1.g: Đồ thị đổi sống biển g(email) Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đổi sống biển h(password):

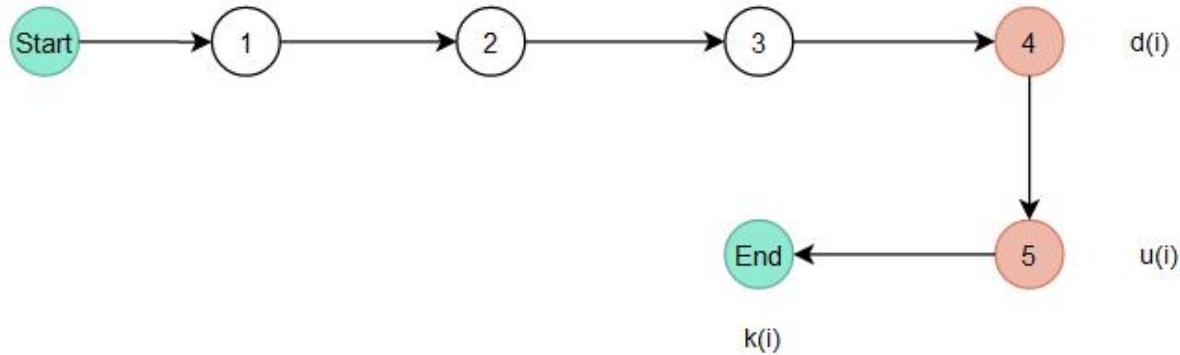


Hình 6.1.h: Đồ thị đổi sống biển h(password) Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biên $i(user)$:



Hình 6.1.i: Đồ thị đời sống biên $i(user)$ Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

2. User - Login

```

//Login User
exports.loginUser = catchAsyncError(async (req, res, next)/*(1)*/
=> {
  const { email, password } = req.body; /*(2)*/

  if (!email || !password)/*(3)*/ {
    return next(new ErrorHander("Please enter email and password", 400)); /*(4)*/
  }

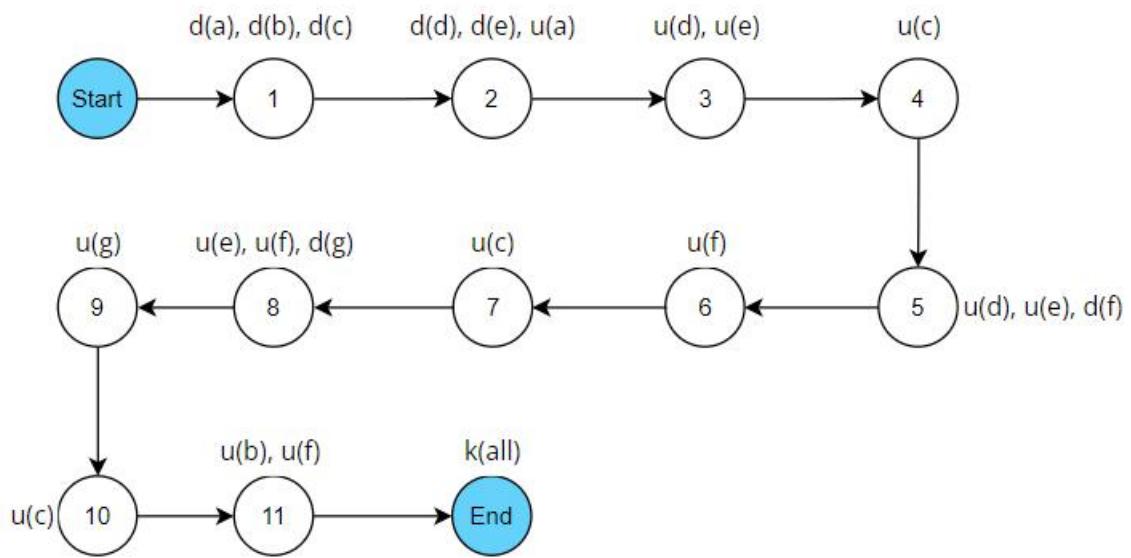
  const user = await User.findOne({ email }).select("+password"); /*(5)*/
  if (!user /*(6)*/) {
    return next(new ErrorHander("Invalid email or password", 401)); /*(7)*/
  }

  const isPasswordMatched = await user.comparePassword(password); /*(8)*/
  if (!isPasswordMatched /*(9)*/) {
    return next(new ErrorHander("Invalid email or password", 401)); /*(10)*/
  }

  sendToken(user, 200, res); /*(11)*/

});
  
```

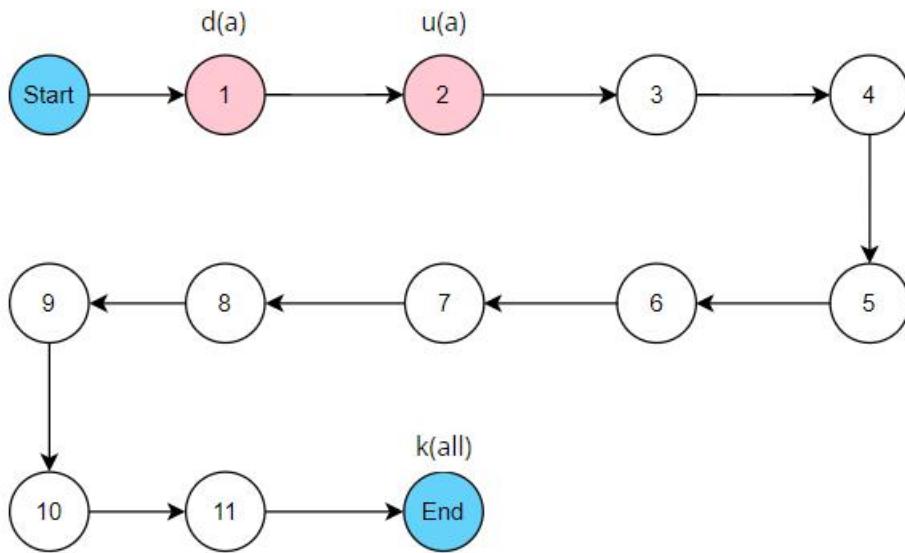
Table 6.2: Code Backend Đăng nhập tài khoản



Hình 6.2: Đồ thị dòng dữ liệu Đăng nhập tài khoản

Kiểm thử đòi hỏi 7 biến: $a(\text{req})$, $b(\text{res})$, $c(\text{next})$, $d(\text{email})$, $e(\text{password})$, $f(\text{user})$, $g(\text{isPasswordMatched})$

- Kiểm thử đòi hỏi biến $a(\text{req})$:

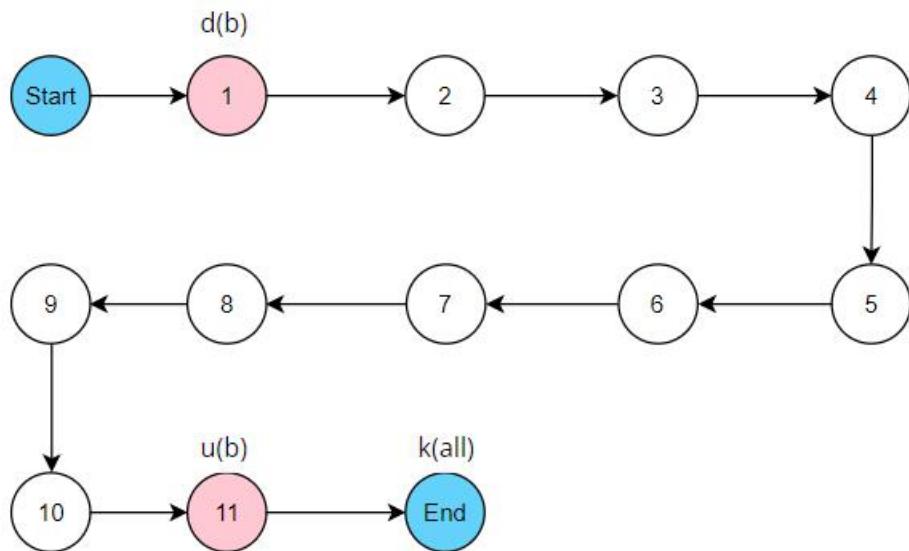


Hình 6.2.a: Đồ thị đòi hỏi biến $a(\text{req})$ Đăng nhập tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển b(res):**

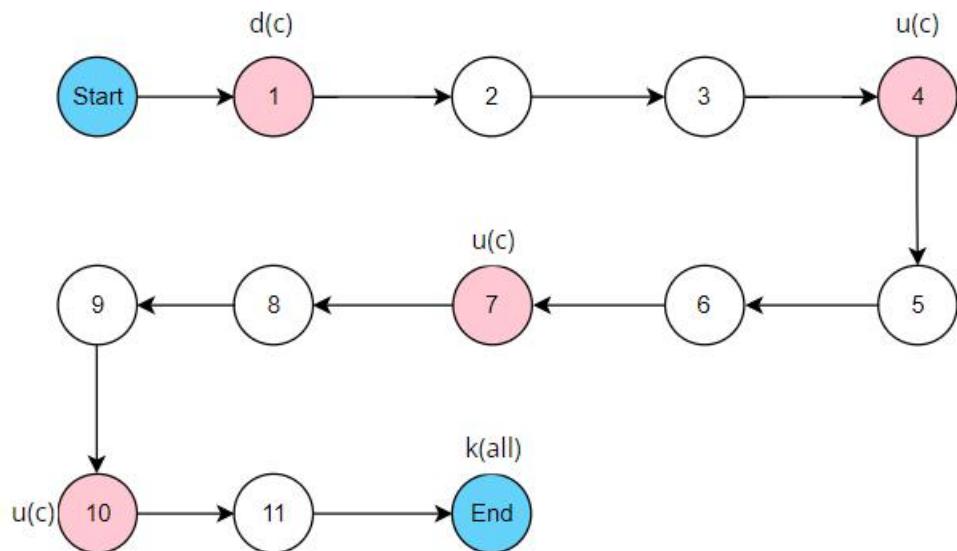


Hình 6.2.b: Đồ thị đời sống biển b(res) Đăng nhập tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển c(next):**



Hình 6.2.c: Đồ thị đời sống biển c(next) Đăng nhập tài khoản

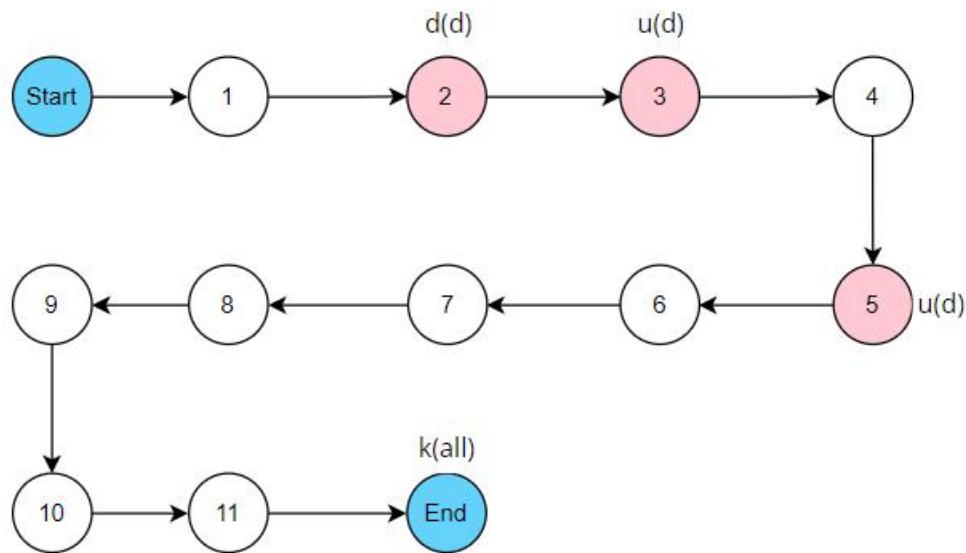
Kịch bản 1: ~duk

Kịch bản 2: ~duuuk

Kịch bản 3: ~duuuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đòi sống biển d(email):**



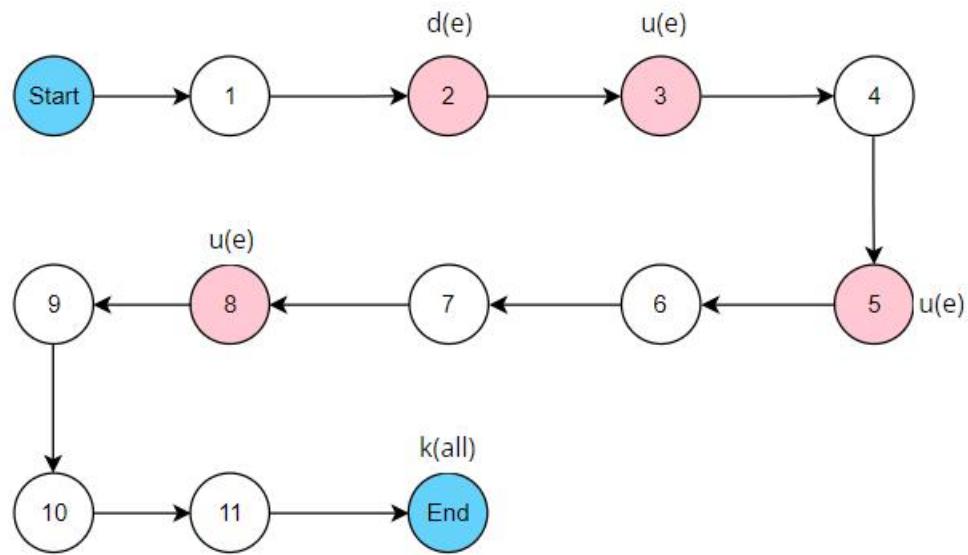
Hình 6.2.d: Đồ thị đòi sống biển d(email) Đăng nhập tài khoản

Kịch bản 1: ~duk

Kịch bản 2: ~duuuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường

- Kiểm thử đời sống biển e(password):



Hình 6.2.e: Đồ thị đời sống biển e(password) Đăng nhập tài khoản

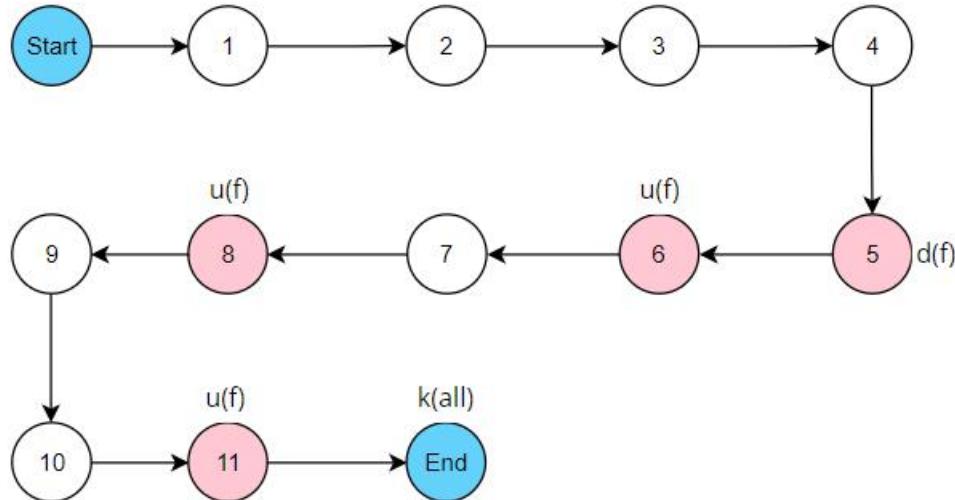
Kịch bản 1: ~duk

Kịch bản 2: ~duuk

Kịch bản 3: ~duuuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường

- Kiểm thử đời sống biển f(user):



Hình 6.2.f: Đồ thị đòn hồi sóng biến $f(user)$ Đăng nhập tài khoản

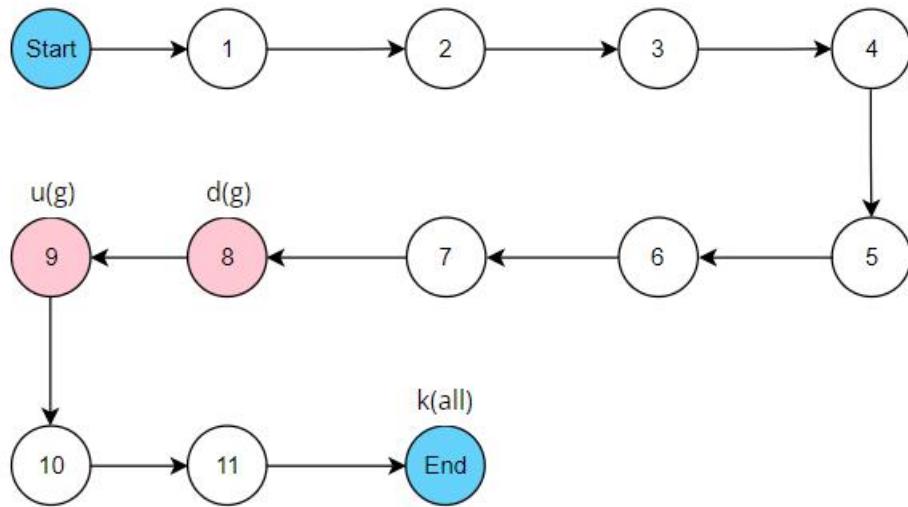
Kịch bản 1: ~duk

Kịch bản 2: ~duuk

Kịch bản 3: ~duuuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường

- Kiểm thử đòn hồi sóng biến $g(isPasswordMatched)$:



Hình 6.2.g: Đồ thị đòn hồi sóng biến $g(isPasswordMatched)$ Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường

3. User - Forgot Password

```

//Forgot Password
exports.forgotPassword = catchAsyncError(async (req, res, next) => {
  const user = await User.findOne({ email: req.body.email });
  if (!user) {
    return next(new ErrorHandler("User not found", 404));
  }

  const resetToken = user.getResetPasswordToken();
  await user.save({ validateBeforeSave: false });

  const resetPasswordUrl = `${req.protocol}://${req.get(
    'host'
  )}/api/users/reset-password/${resetToken}`;
  const message = `Your password reset token is valid for 10 minutes. Your new password will be required to log in.\n\n${resetPasswordUrl}`;
  mailer.sendMail({
    to: user.email,
    subject: "Forgot Password",
    template: "reset",
    context: { name: user.name, message }
  });
  res.status(200).json({
    success: true,
    message: "Email sent to reset password"
  });
});
  
```

```

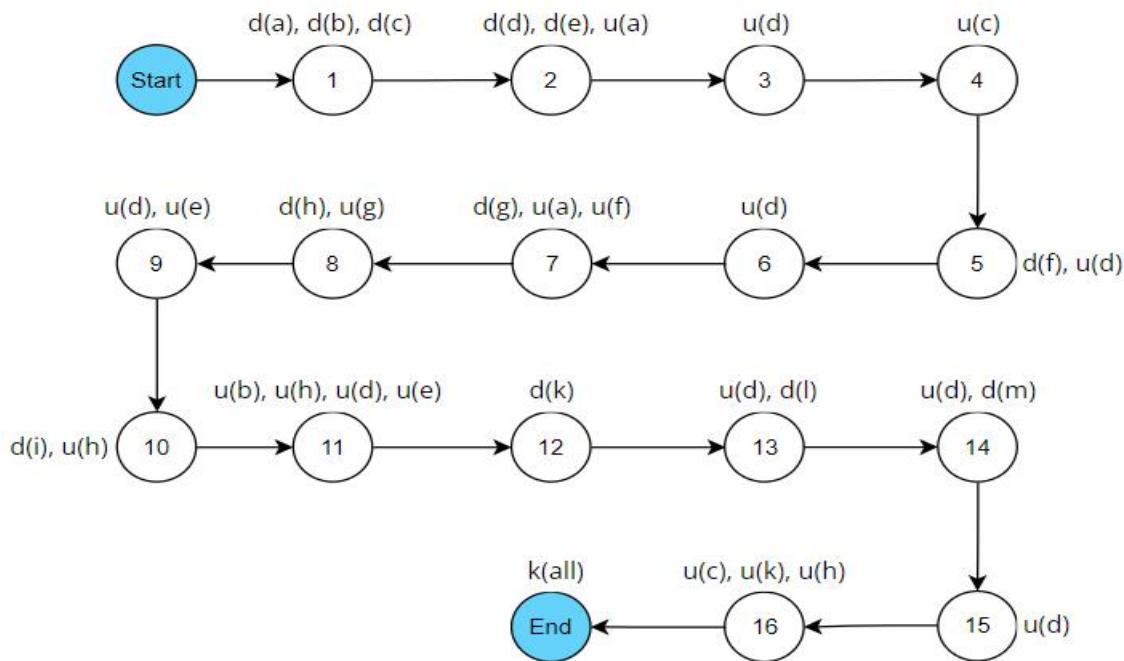
    "host"
})/api/v1/password/reset/${resetToken}; /*(7)*/

const message = `Your password reset token is temporary: - \n\n
${resetPasswordUrl} \n\nIf you have not requested this email then, please
ignore it.`; /*(8)*/
try /*(9)*/ {
    await sendEmail({
        email: user.email,
        subject: `Website Password Recovery`, /*(10)*/,
        message,
    });

    res.status(200).json({
        success: true, /*(11)*/,
        message: `Email sent to ${user.email} successfully`,
    });
} catch (error/*(12)*i) {
    user.resetPasswordToken = undefined; /*(13)*/,
    user.resetPassordExpire = undefined; /*(14)*/,
    await user.save({ validateBeforeSave: false }); /*(15)*/,
    return next(new ErrorHander(error.message, 500)); /*(16)*/,
}
}
);

```

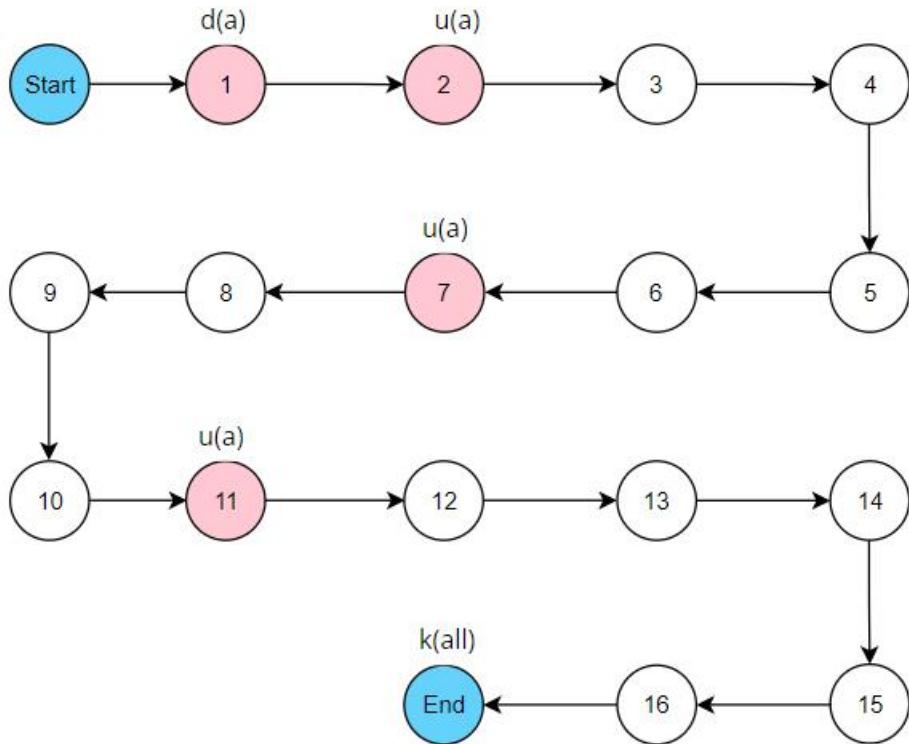
Table 6.3: Code Backend Quên mật khẩu



Hình 6.3: Đồ thị dòng dữ liệu Quên mật khẩu

Kiểm thử đời sống 12 biến: $a(req)$, $b(res)$, $c(next)$, $d(user)$, $e(email)$, $f(resetToken)$,
 $g(resetPasswordUrl)$, $h(message)$, $i(subject)$, $k(error)$, $l(resetPasswordToken)$,
 $m(resetPasswordExpire)$

- **Kiểm thử đời sống biến $a(req)$:**



Hình 6.3.a: Đồ thị đời sống biến $a(req)$ Quên mật khẩu

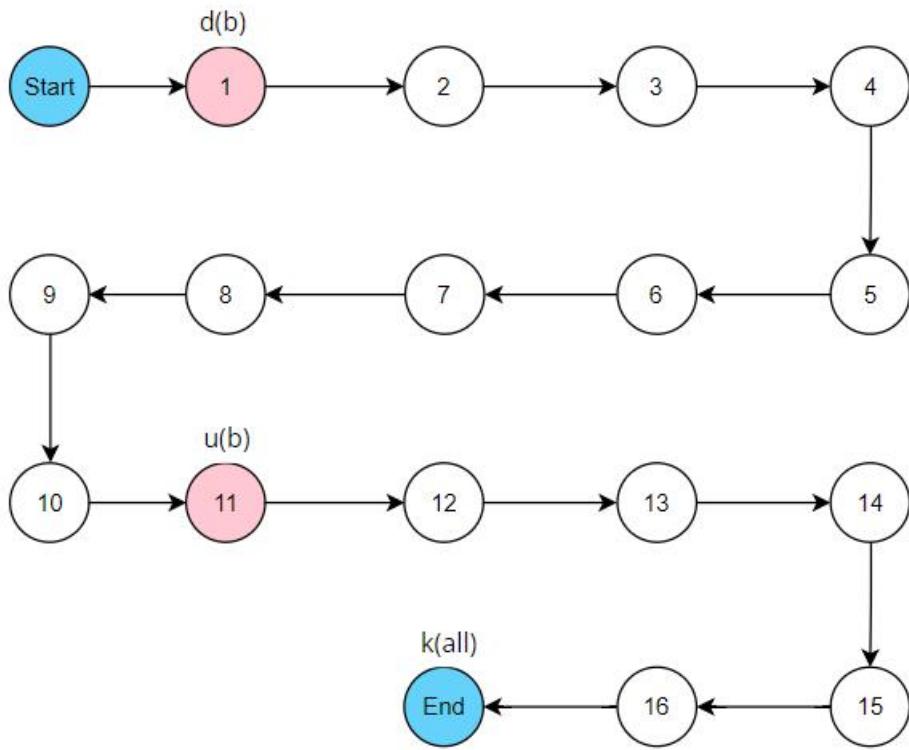
Kịch bản 1: ~duk

Kịch bản 2: ~duuuk

Kịch bản 3: ~duuuuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đòn bẩy biên $b(res)$:

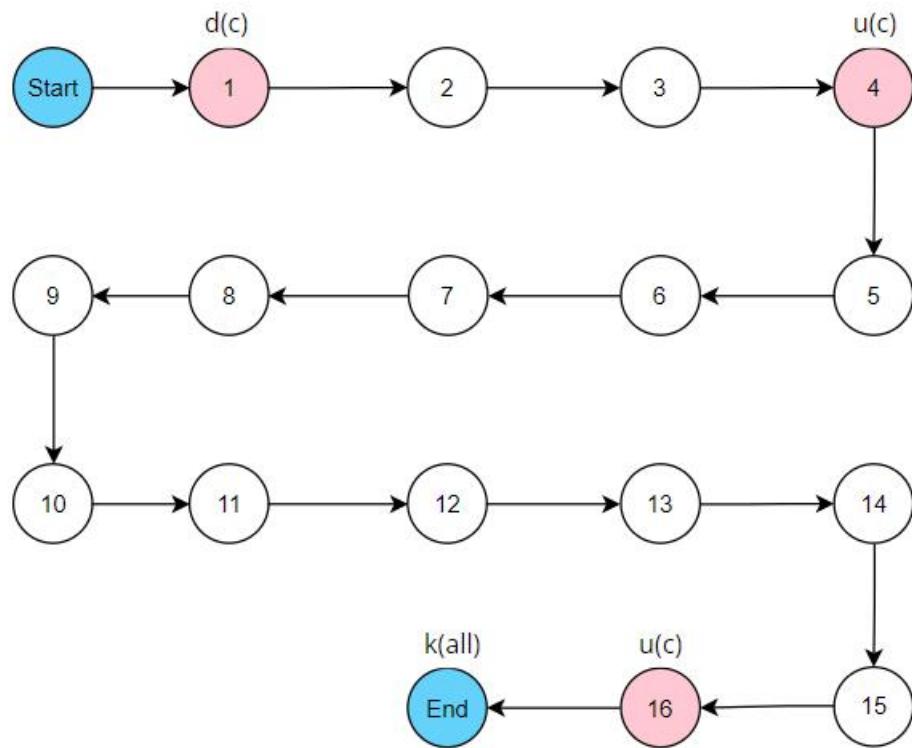


Hình 6.3.b: Đồ thị đòn bẩy biên $b(res)$ Quên mật khẩu

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đòn tảng biên $c(next)$:



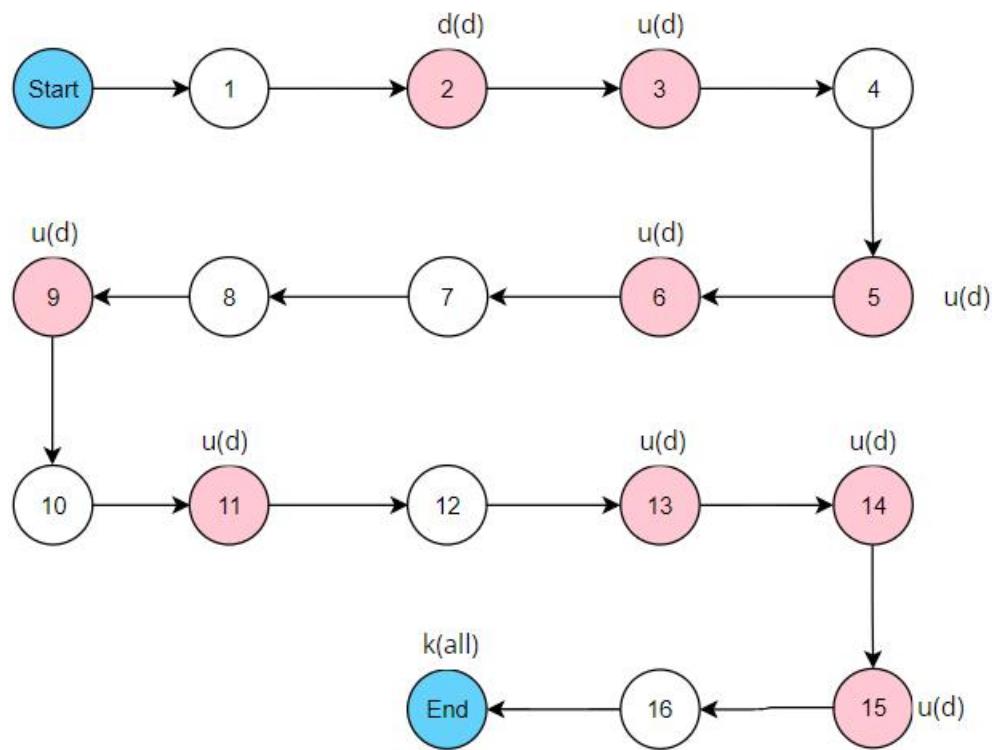
Hình 6.3.c: Đồ thị đòn tảng biên $c(next)$ Quên mật khẩu

Kịch bản 1: ~duk

Kịch bản 2: ~duuuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sống biển $d(user)$:



Hình 6.3.d: Đồ thị đời sống biển $d(user)$ Quên mật khẩu

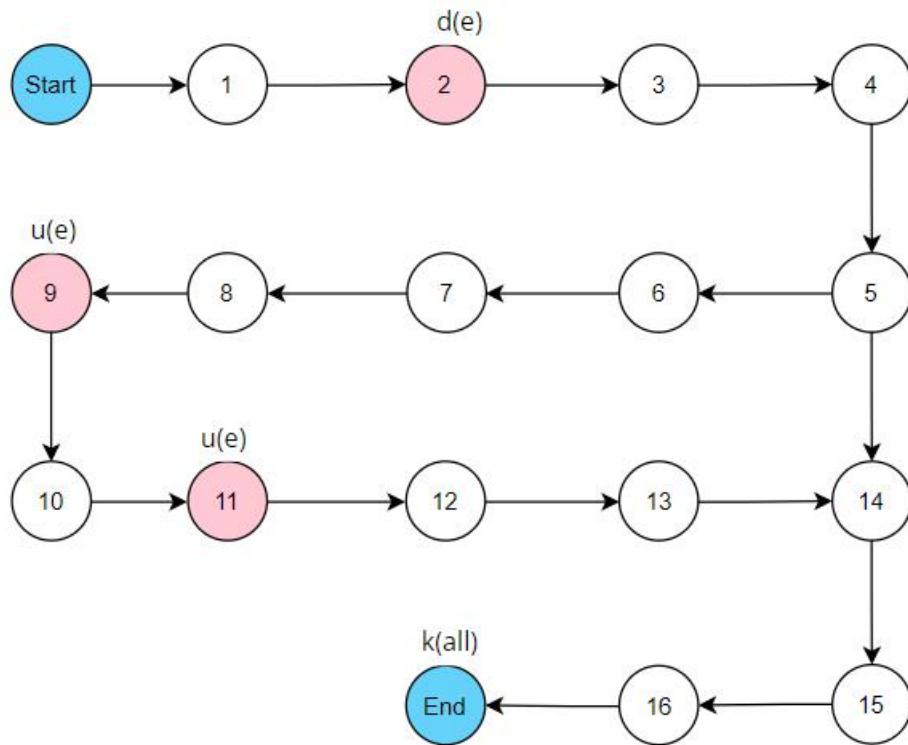
Kịch bản 1: ~duk

Kịch bản 2: ~duuuk

Kịch bản 3: ~duuuuuuuuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sống biển e(email):



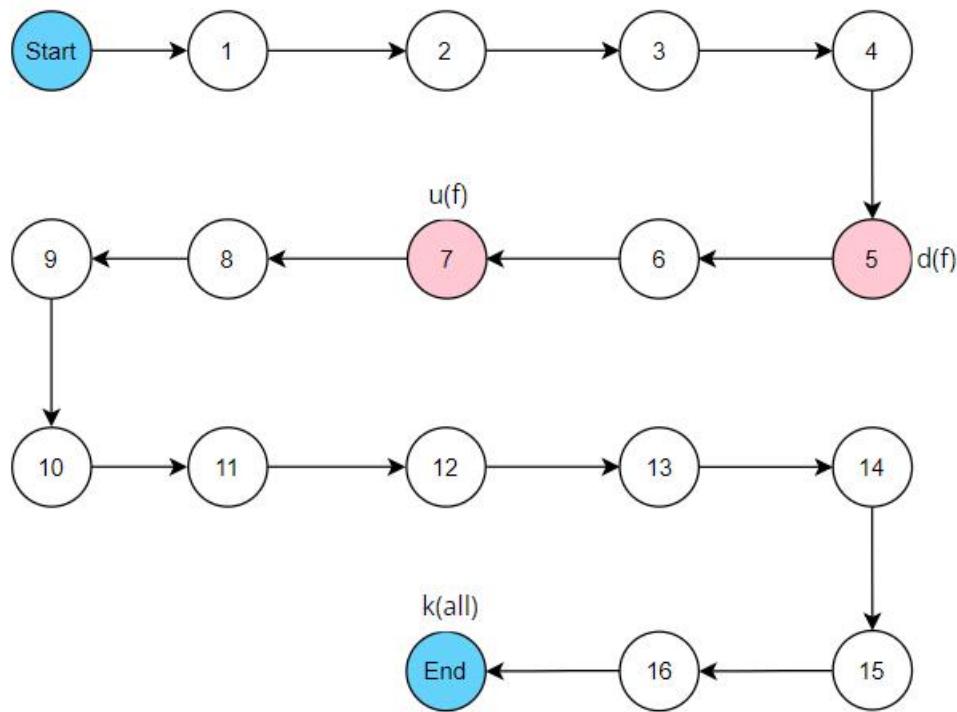
Hình 6.3.e: Đồ thị đời sống biển e(email) Quên mật khẩu

Kịch bản 1: ~duk

Kịch bản 2: ~duuk

=> **Kết luận:** Không có cặp đôi nào của hoạt động bất thường.

- Kiểm thử đòi sống biến f (resetToken):

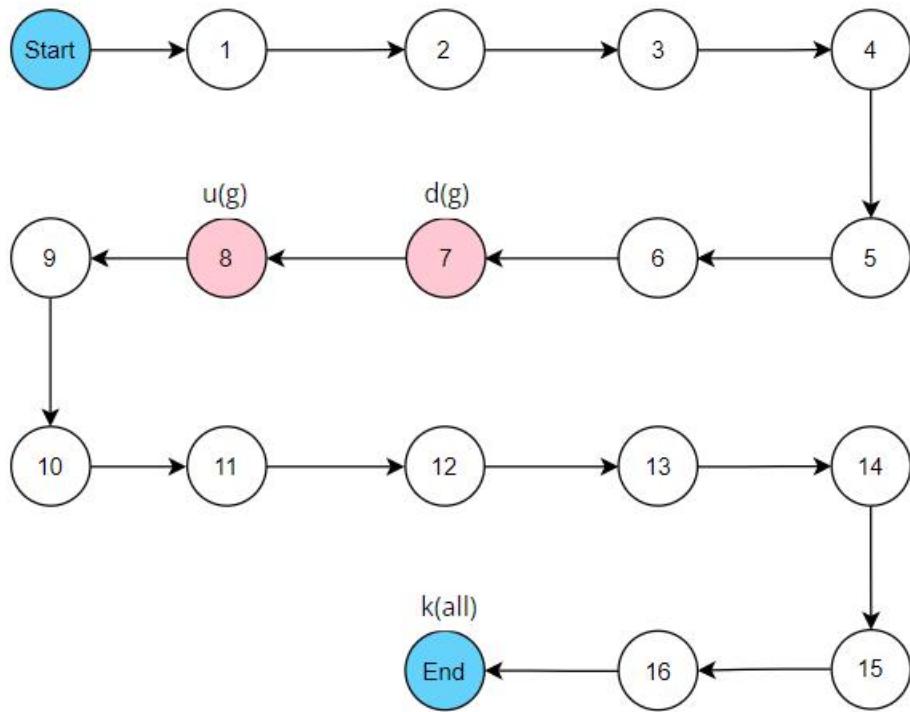


Hình 6.3.f: Đồ thị đòi sống biến f (resetToken) Quên mật khẩu

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của hoạt động bất thường.

- Kiểm thử đối số biến g(resetPasswordUrl):

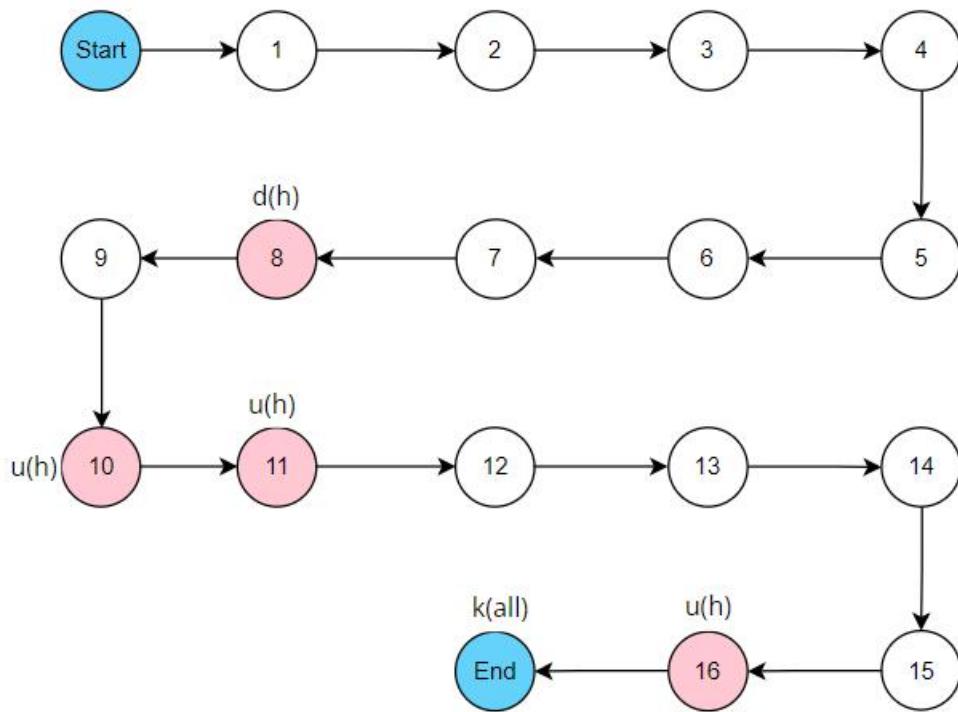


Hình 6.3.g: Đồ thị đối số biến g(resetPasswordUrl) Quên mật khẩu

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biến h (message):



Hình 6.3.h: Đồ thị đời sóng biến h (message) Quên mật khẩu

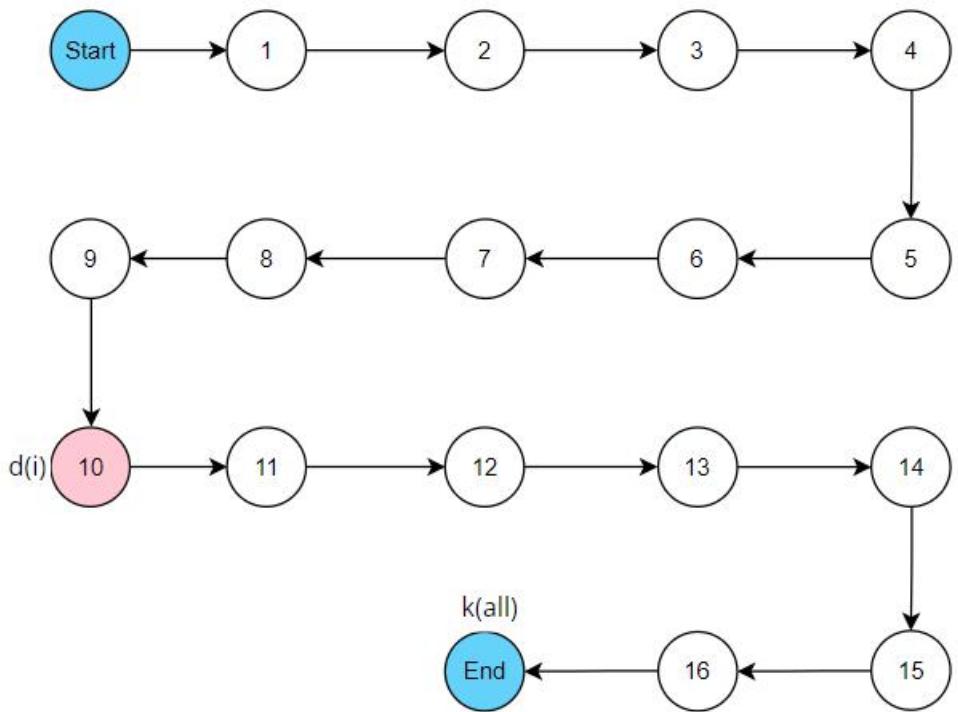
Kịch bản 1: ~duk

Kịch bản 2: ~duuk

Kịch bản 3: ~duuk

=> **Kết luận:** Không có cặp đôi nào của hoạt động bất thường.

- Kiểm thử đời sống biển i(subject):

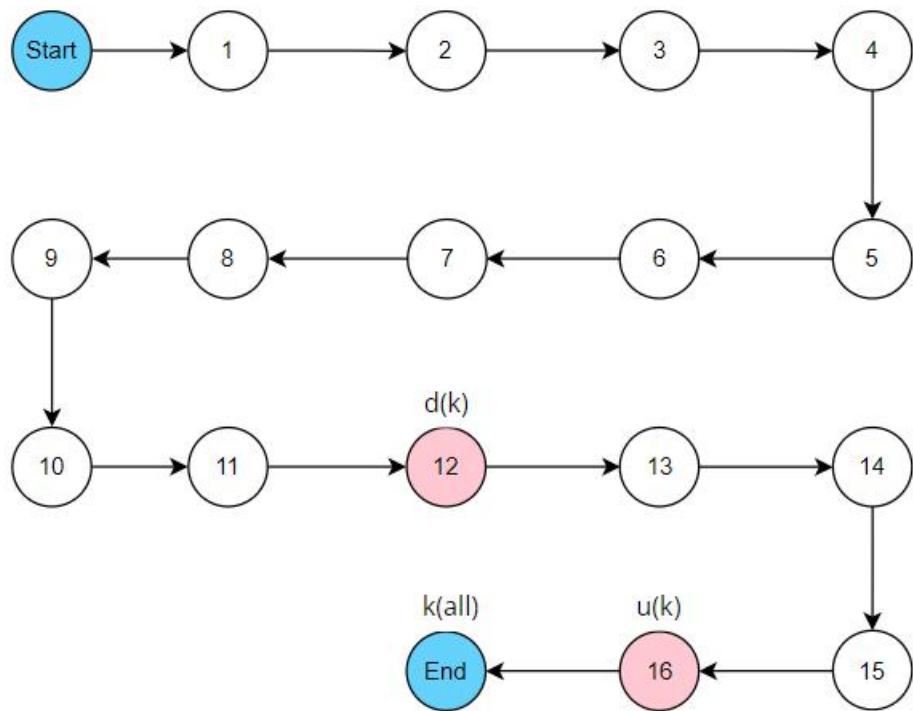


Hình 6.3.i: Đồ thị đời sống biển i(subject) Quên mật khẩu

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đòn sóng biên k(error):

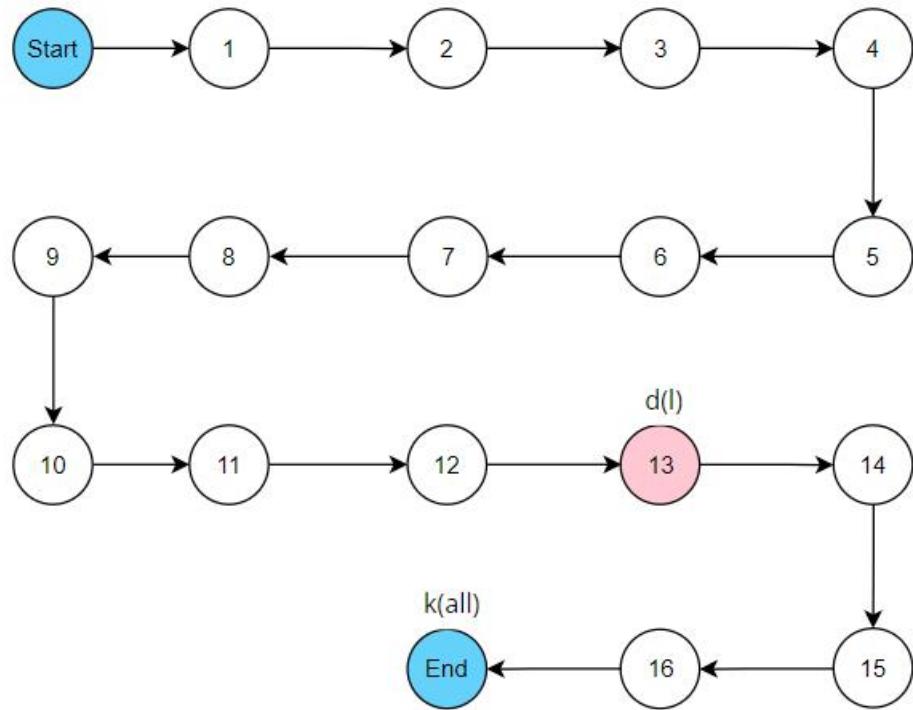


Hình 6.3.k: Đồ thị đòn sóng biên k(error) Quên mật khẩu

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đòi hỏi biến l(*resetPasswordToken*):

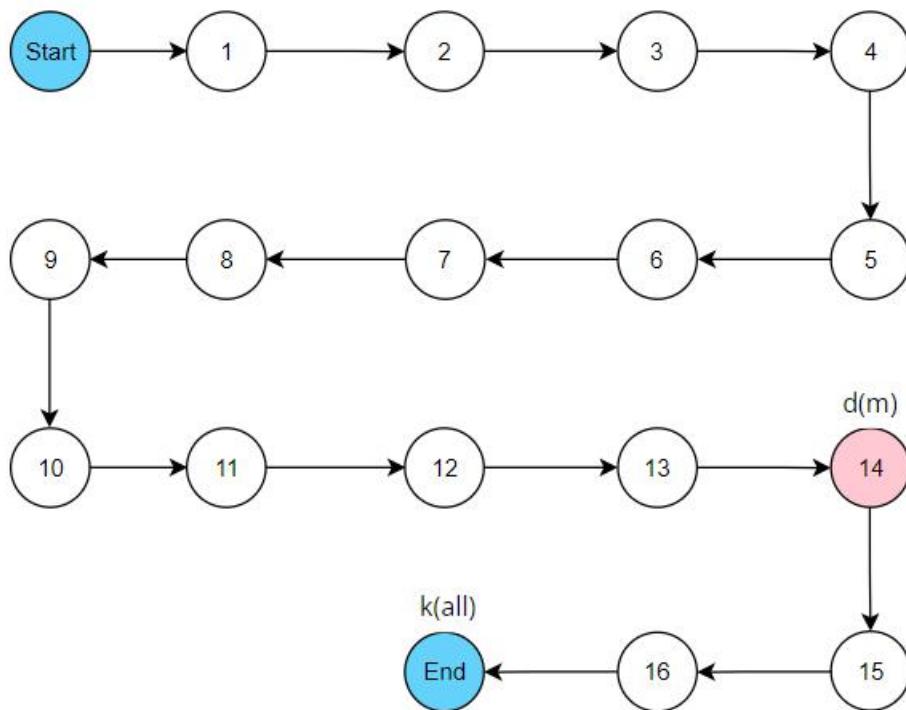


Hình 6.3.l: Đồ thị đòi hỏi biến l(*resetPasswordToken*) Quên mật khẩu

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đòi hỏi biên m(*resetPasswordExpire*):



Hình 6.3.m: Đồ thị đòi hỏi biên m(*resetPasswordExpire*) Quên mật khẩu

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

4. User - Reset Password

```

//Reset Password
exports.resetPassword = catchAsyncError(async (req, res, next/*(1)*/) => {
  const resetPasswordToken = crypto
    .createHash("sha256")
    .update(req.params.token)
    .digest("hex"); /*(2)*/

  const user = await User.findOne({
    resetPasswordToken,
    resetPasswordExpire: { $gt: Date.now() },
  }); /*(3)*/

  if (!user /*(4)*/ ) {
    return next(
      new ErrorHander(
        "User with this token does not exist"
      )
    );
  }

  const resetPasswordHash = crypto
    .createHash("sha256")
    .update(user.resetPasswordToken)
    .digest("hex");
  user.resetPasswordHash = resetPasswordHash;
  user.resetPasswordExpire = Date.now() + 10 * 60 * 1000;
  user.save();
  res.status(200).json({ message: "Reset password token generated successfully" });
}
  
```

```

    "Reset Password Token is invalid or has been expired",
    400
)
);
/*(5)*/
}

if (req.body.password !== req.body.confirmPassword /*(6)*/
{
    return next(new ErrorHander("Password does not match", 400)); /*(7)*/
}

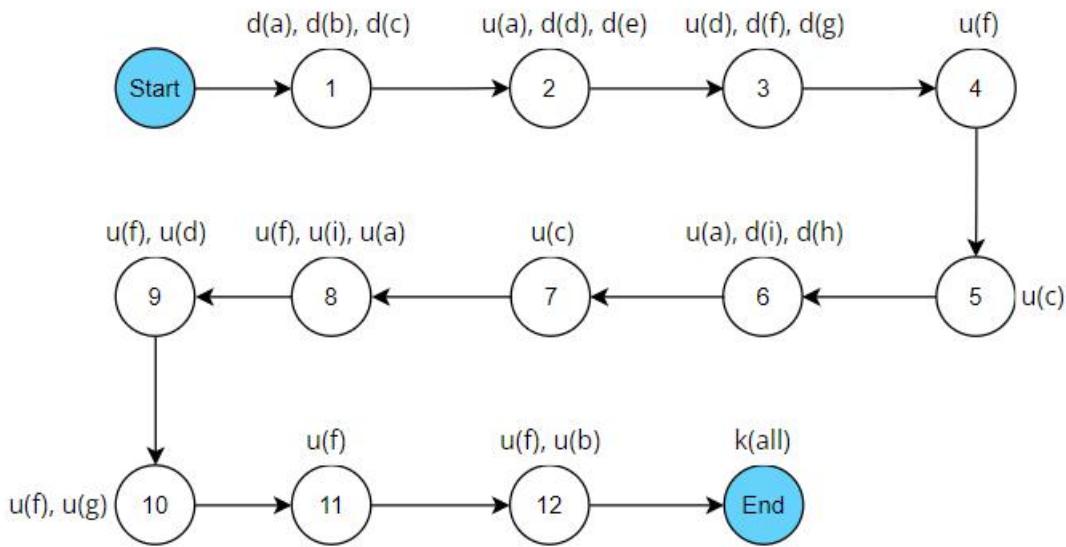
user.password = req.body.password; /*(8)*/
user.resetPasswordToken = undefined; /*(9)*/
user.resetPasswordExpire = undefined; /*(10*/

await user.save(); /*(11)*

sendToken(user, 200, res); /*(12)*/
});

```

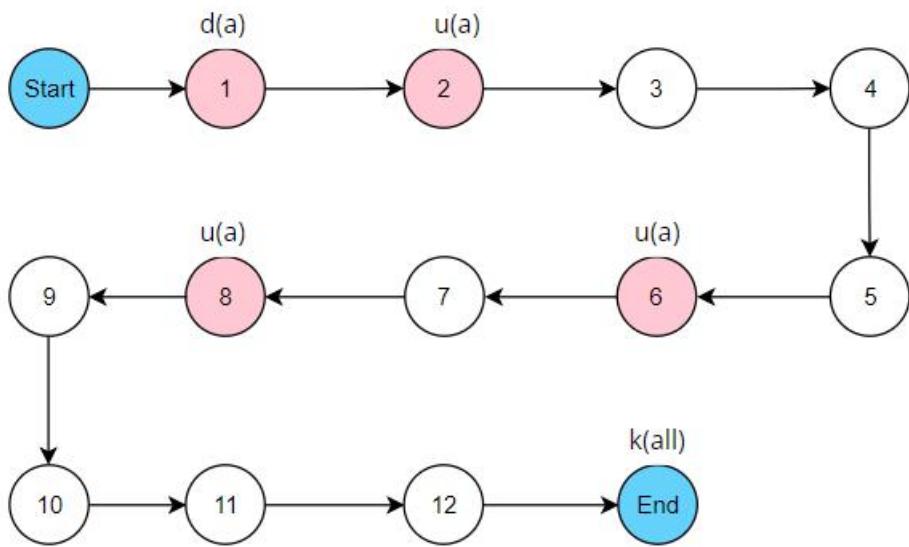
Table 6.4: Code Backend Cài lại mật khẩu



Hình 6.4: Đồ thị dòng dữ liệu Cài lại mật khẩu

Kiểm thử đòn hồi sóng 9 biến: a(req), b(res), c(next), d(resetPasswordToken), e(token), f(user), g(resetPasswordExpire), h(confirmPassword), i(password)

- Kiểm thử đời sóng biển a(req):



Hình 6.4.a: Đồ thị đời sóng biển a(req) Cài lại mật khẩu

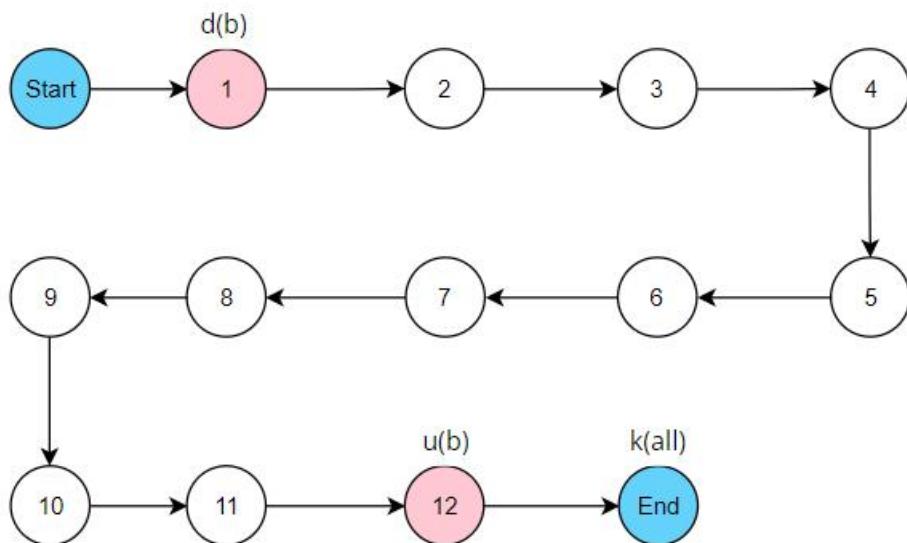
Kịch bản 1: ~duk

Kịch bản 2: ~duuk

Kịch bản 3: ~duuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sóng biển b(res):

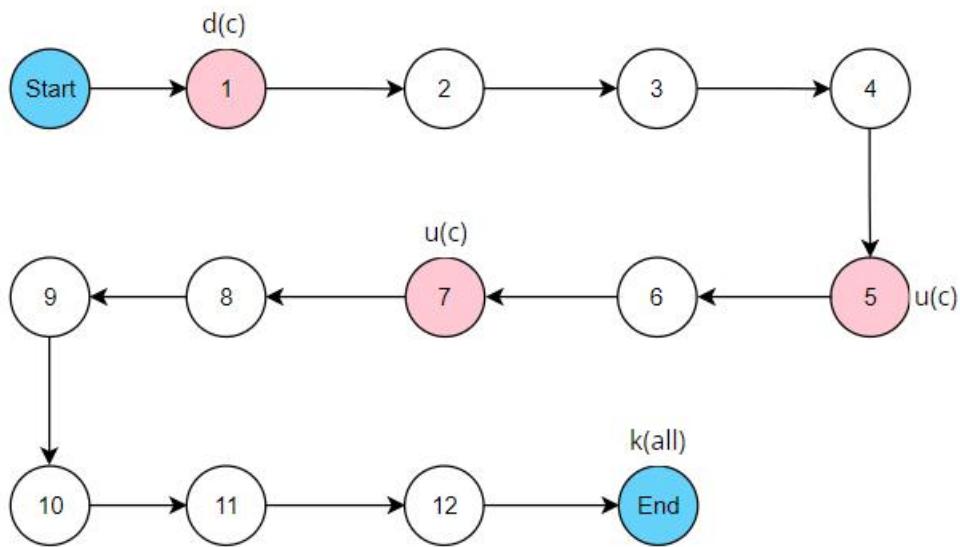


Hình 6.4.b: Đồ thị đòn sóng biển b(res) Cài lại mật khẩu

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đòn sóng biển c(next):



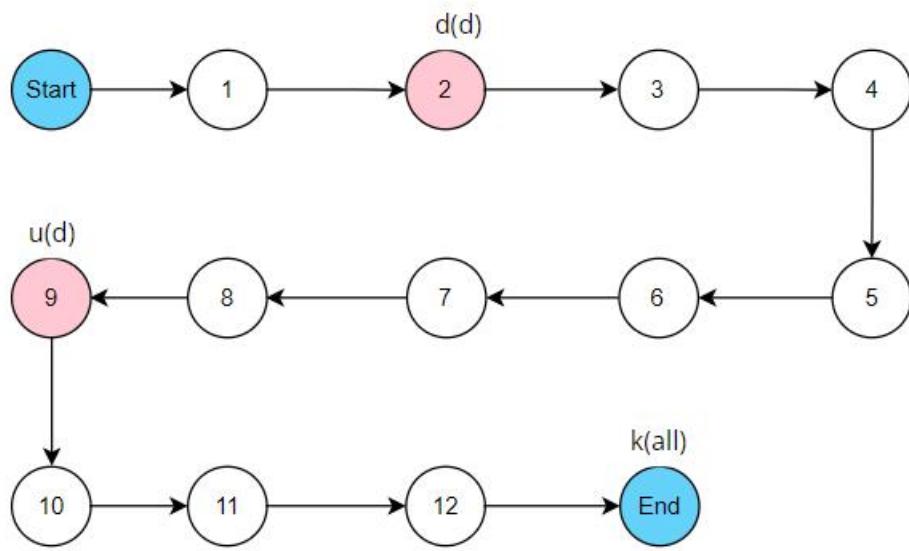
Hình 6.4.c: Đồ thị đòn sóng biển c(next) Cài lại mật khẩu

Kịch bản 1: ~duk

Kịch bản 2: ~duuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đòi hỏi biến $d(resetPasswordToken)$:

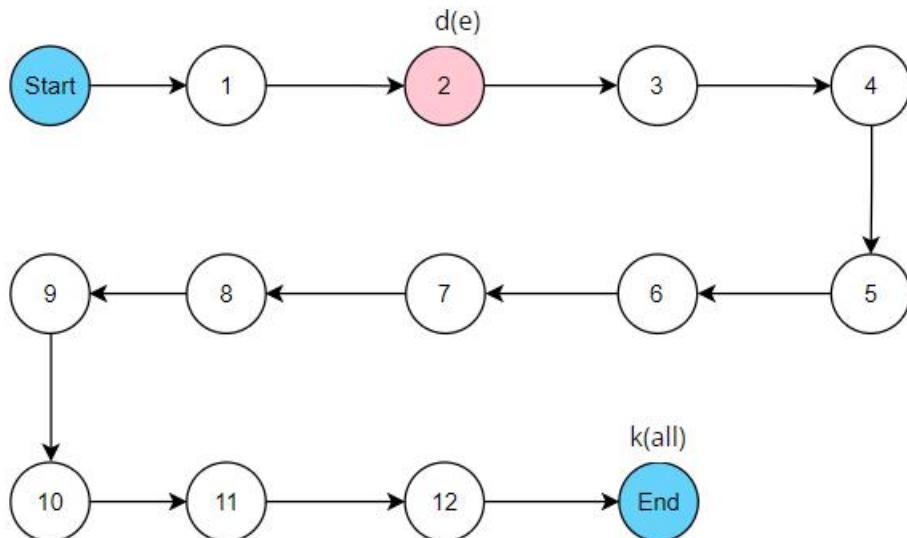


Hình 6.4.d: Đồ thị đòi hỏi biến $d(resetPasswordToken)$ Cài lại mật khẩu

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đòi hỏi biến $e(token)$:

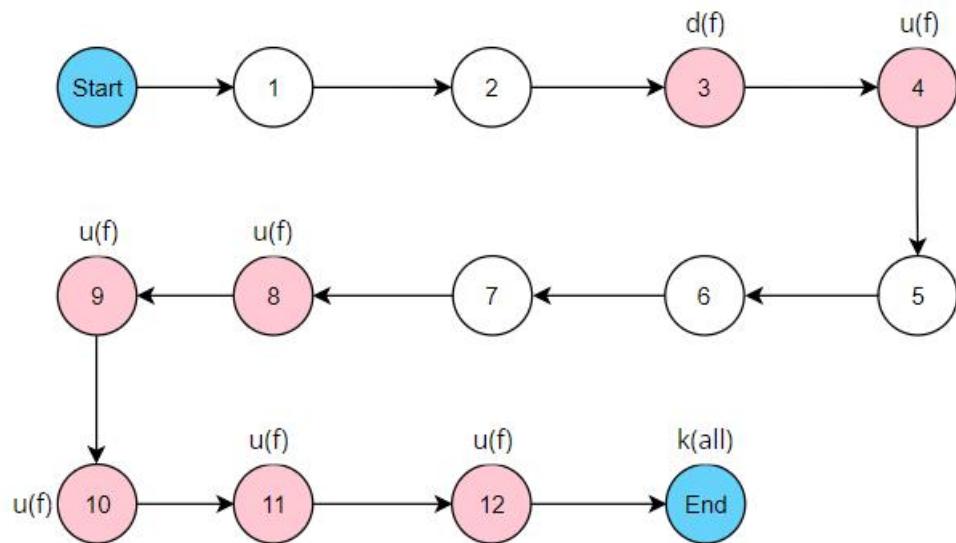


Hình 6.4.e: Đồ thị đòi hỏi biến $e(token)$ Cài lại mật khẩu

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển f(user):**



Hình 6.4.f: Đồ thị đời sống biển f(user) Cài lại mật khẩu

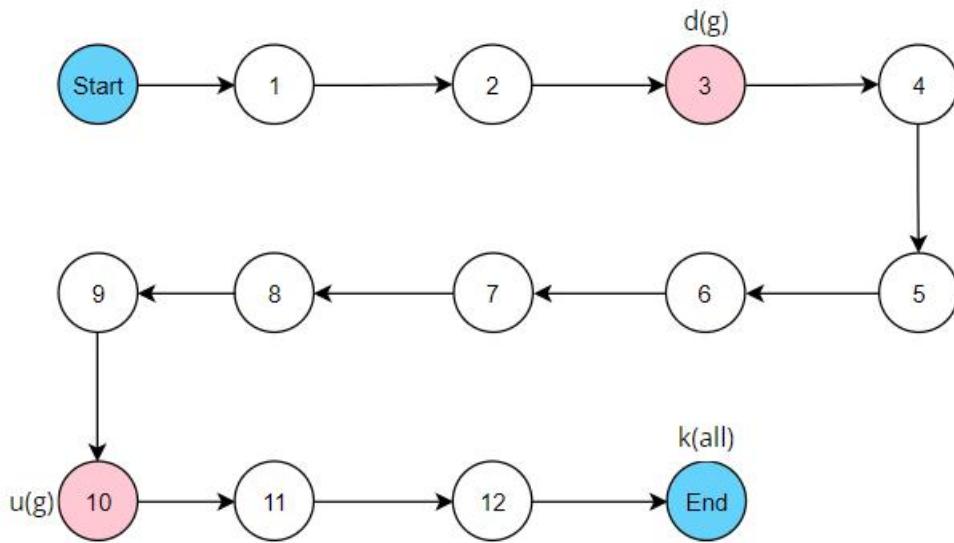
Kịch bản 1: ~duk

Kịch bản 2: ~duuuk

Kịch bản 3: ~duuuuuuk

=> **Kết luận:** Không có cặp đôi nào của hoạt động bất thường.

- Kiểm thử đời sống biến g (resetPasswordExpire):

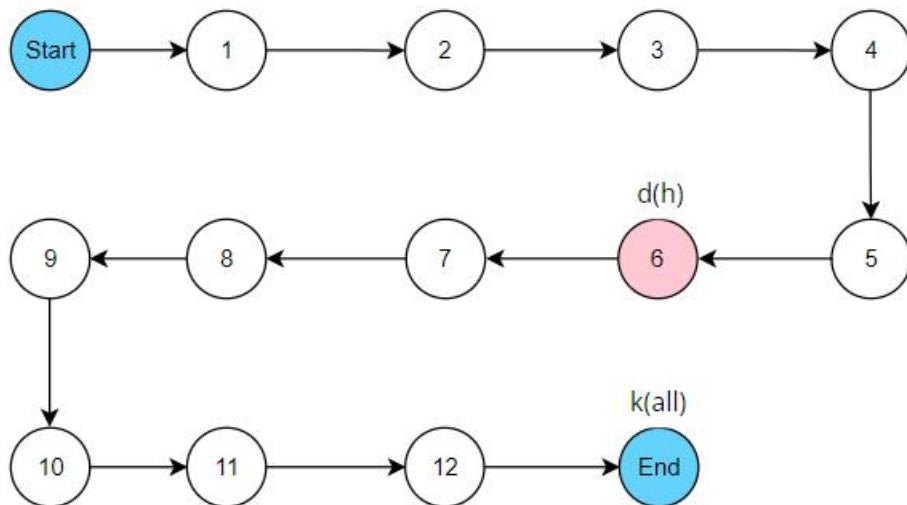


Hình 6.4.g: Đồ thị đời sống biến g (resetPasswordExpire) Cài lại mật khẩu

Kịch bản 1: ~duk

=> Kết luận: Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến h (confirmPassword):

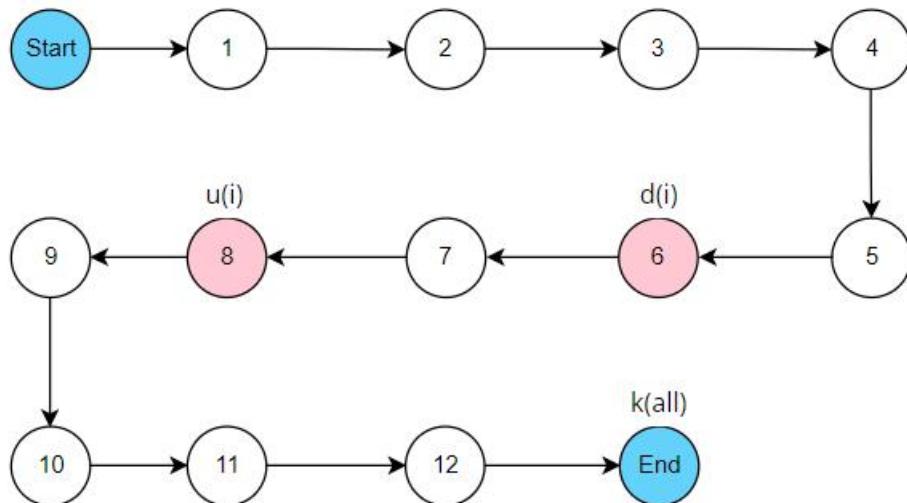


Hình 6.4.h: Đồ thị đời sống biến h (confirmPassword) Cài lại mật khẩu

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đổi sóng biển i(password):**



Hình 6.4.i: Đồ thị đổi sóng biển i(password) Cài lại mật khẩu

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

5. User - See All Products

```
//Get All Products
exports.getAllProducts = catchAsyncError(async (req, res, next/*(1)*)) => {
  const resultPerPage = 8; /*(2)*/
  const productsCount = await Product.countDocuments(); /*(3)*/

  const apiFeature = new ApiFeatures(Product.find(), req.query)
    .search() /*(4)*/
    .filter(); /*(4)*/

  let products = await apiFeature.query; /*(5)*/

  let filteredProductsCount = products.length; /*(6)*/

  apiFeature.pagination(resultPerPage); /*(7)*/

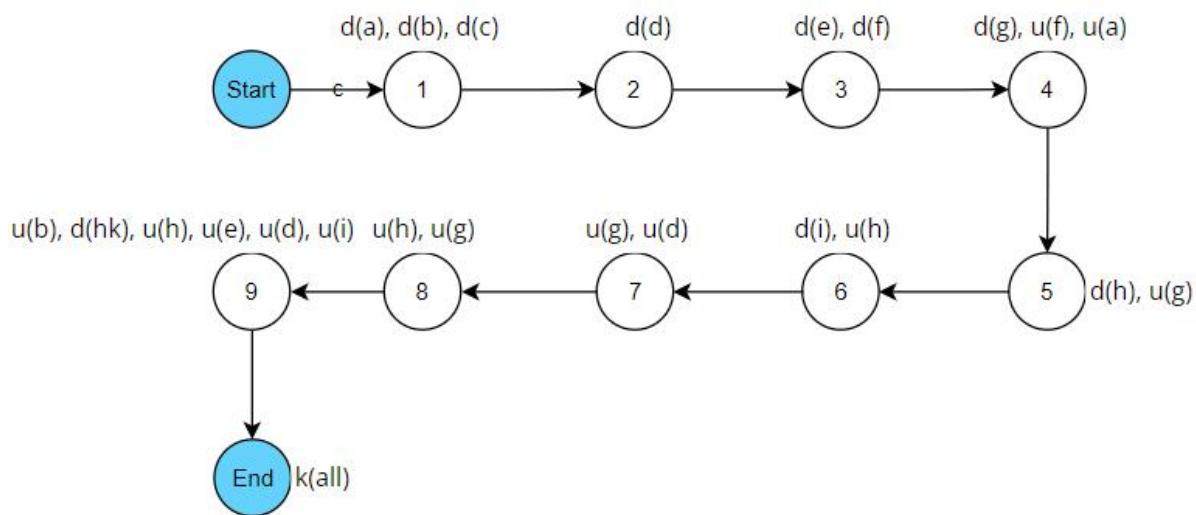
  products = await apiFeature.query.clone(); /*(8)*/
```

```

res.status(200).json({
  success: true,
  products,
  productsCount,
  resultPerPage,
  filteredProductsCount,
}); /*(9)*/
});

```

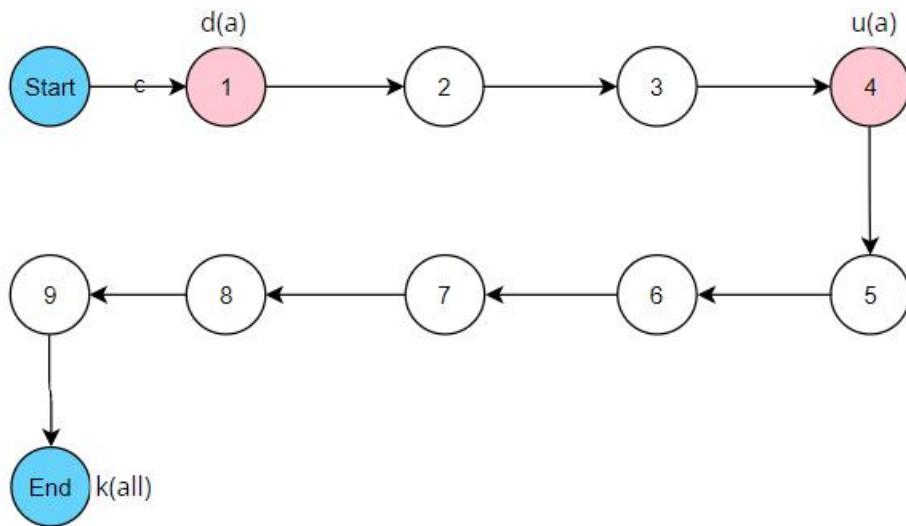
Table 6.5: Code Backend Xem tất cả các sản phẩm



Hình 6.5: Đồ thị dòng dữ liệu Xem tất cả sản phẩm

Kiểm thử dõi sống 10 biến: a(req), b(res), c(next), d(resultPerPage), e(productsCount), f(Product), g(apiFeature), h(products), i(filteredProductsCount), k(success)

- Kiểm thử đời sóng biển a(req):

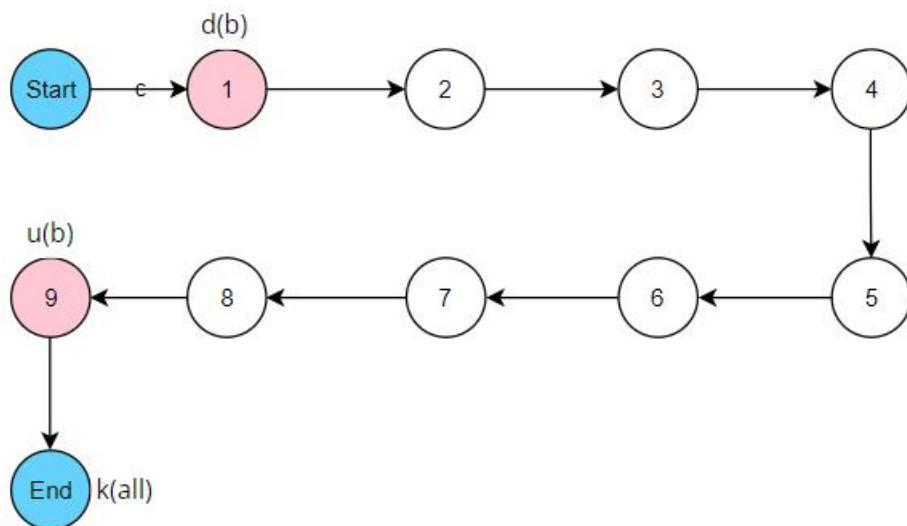


Hình 6.5.a: Đồ thị đời sóng biển a(req) Xem tất cả sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển b(res):

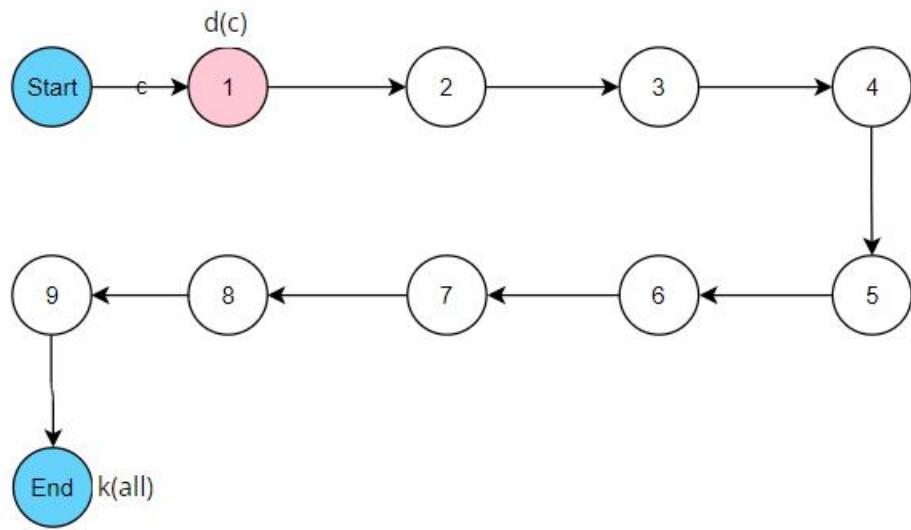


Hình 6.5.b: Đồ thị đời sóng biển b(res) Xem tất cả sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biển $c(next)$:

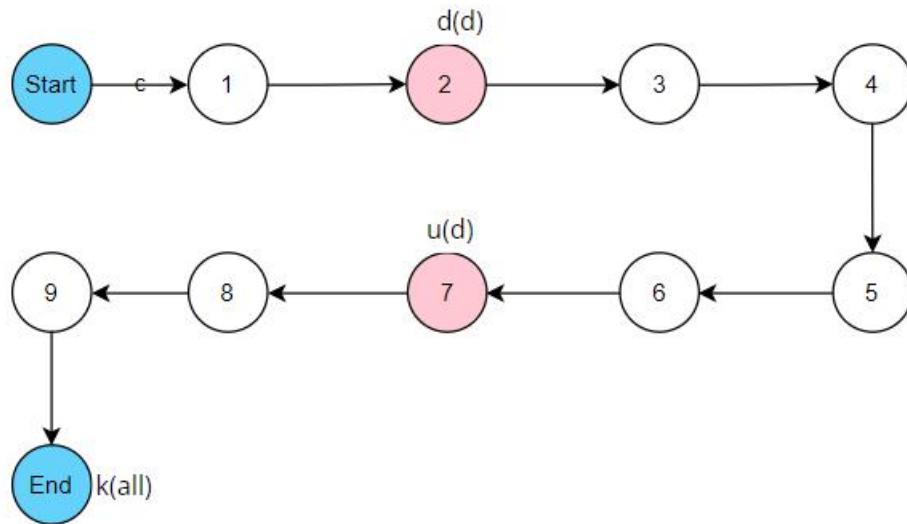


Hình 6.5.c: Đồ thị đời sống biển $c(next)$ Xem tất cả sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biển $d(resultPerPage)$:

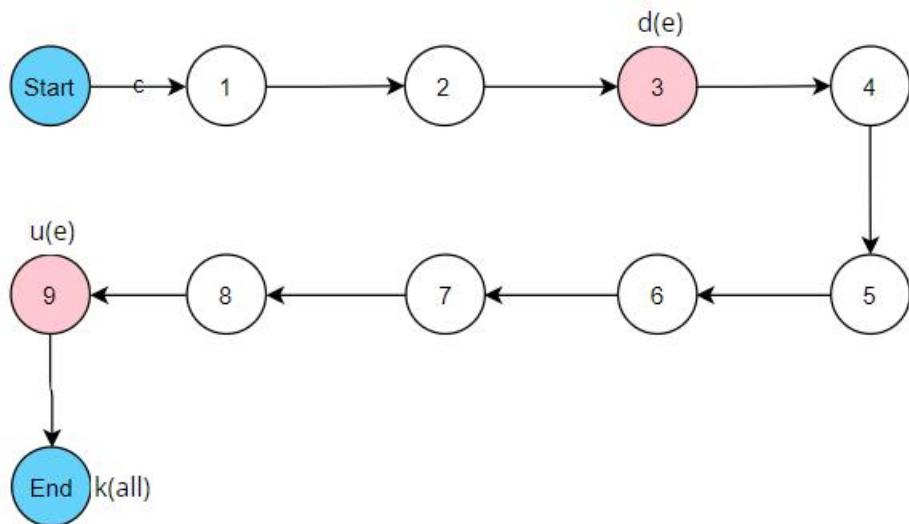


Hình 6.5.d: Đồ thị đời sống biển $d(resultPerPage)$ Xem tất cả sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biển e(productsCount):**

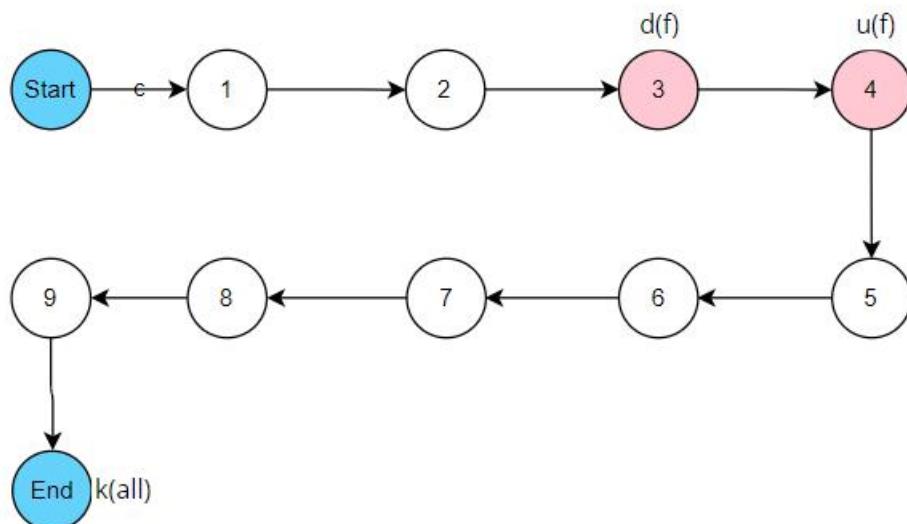


Hình 6.5.e: Đồ thị đời sóng biển e(productsCount) Xem tất cả sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biển f(Product):**

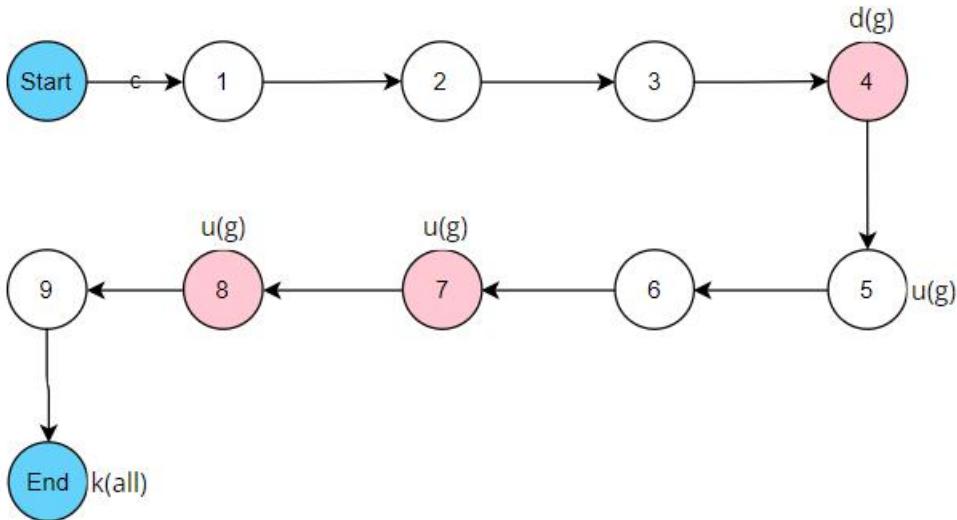


Hình 6.5.f: Đồ thị đời sóng biển f(Product) Xem tất cả sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biển g(apiFeature):**



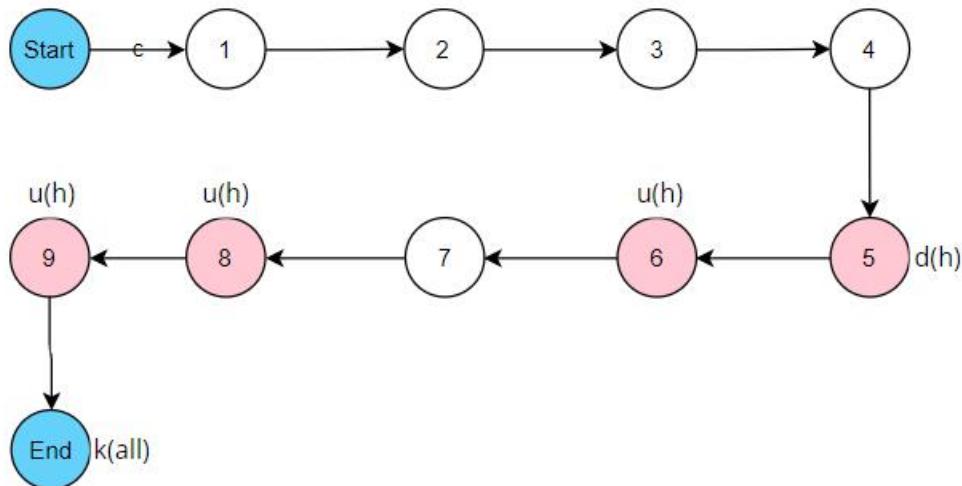
Hình 6.5.g: Đồ thị đời sóng biển g(apiFeature) Xem tất cả sản phẩm

Kịch bản 1: ~duk

Kịch bản 2: ~duuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đời sóng biển h(products):**



Hình 6.5.h: Đồ thị đời sóng biển h(products) Xem tất cả sản phẩm

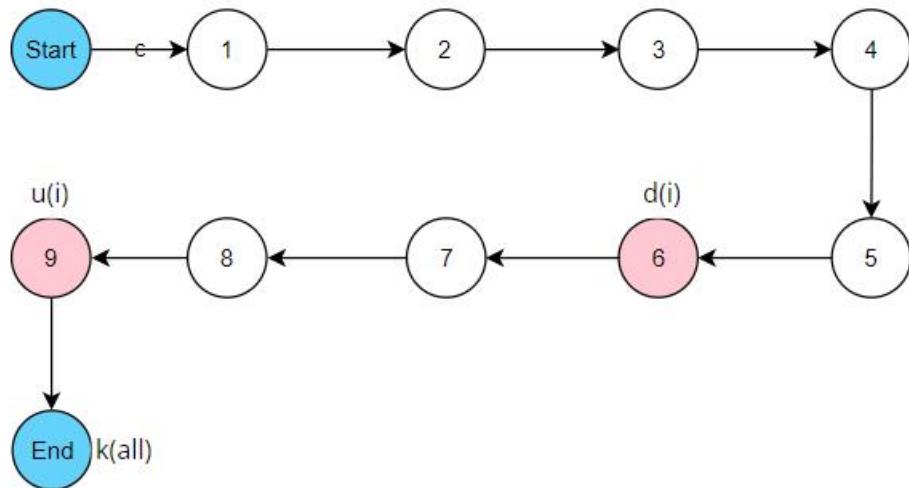
Kịch bản 1: ~duk

Kịch bản 2: ~duuk

Kịch bản 3: ~duuuk

=> **Kết luận:** Không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đòn sóng biển i(filteredProductsCount):**

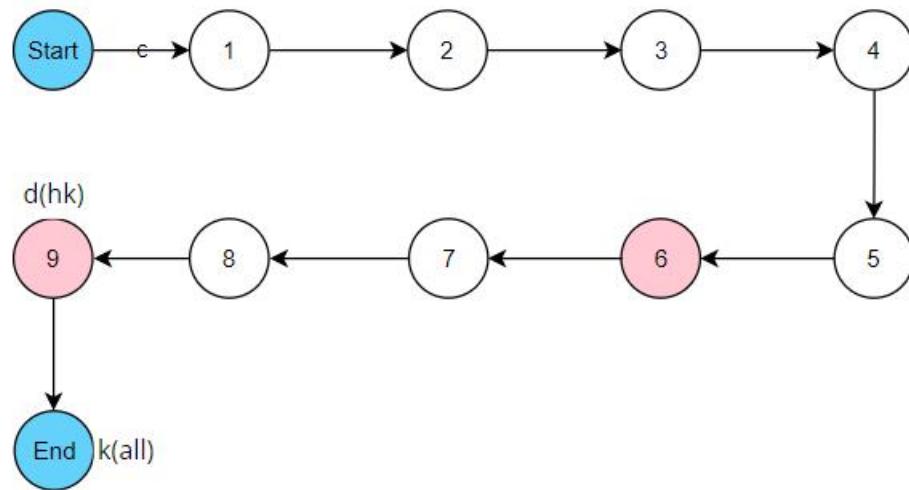


Hình 6.5.i: Đồ thị đòn sóng biển i(filteredProductsCount) Xem tất cả sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biển hk(success):



Hình 6.5.k: Đồ thị đời sống biển k(success) Xem tất cả sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

6. User - See A Product Details

```

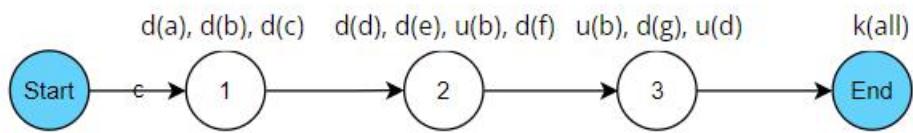
//Get A Product Details
exports.getProductDetails = catchAsyncError(async (req, res, next/*(1)*I) => {
  const product = await Product.findById(req.params.id); /*(2)*I

  if (!product /*(3)*I) {
    return next(new ErrorHander("Product not found", 404)); /*(4)*I
  }

  res.status(200).json({
    success: true,
    product,
  }); /*(5)*I
});

```

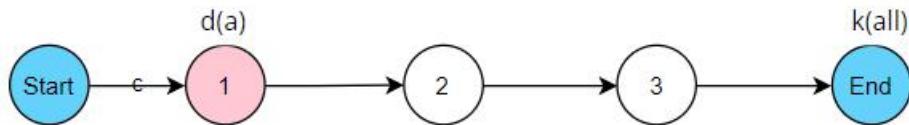
Table 6.6: Code Backend Xem thông tin chi tiết 1 sản phẩm



Hình 6.6: Đồ thị dòng dữ liệu Xem thông tin chi tiết 1 sản phẩm

Kiểm thử đời sống 7 biến: $a(\text{req})$, $b(\text{res})$, $c(\text{next})$, $d(\text{product})$, $e(\text{Product})$, $f(\text{id})$, $g(\text{success})$

- **Kiểm thử đời sống biến $a(\text{req})$:**

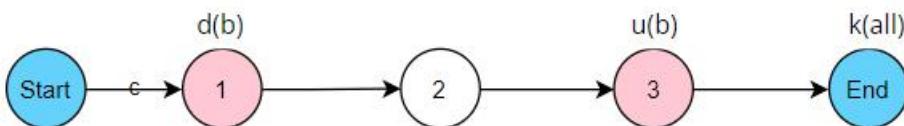


Hình 6.6.a: Đồ thị đời sống biến $a(\text{req})$ Xem thông tin chi tiết 1 sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biến $b(\text{res})$:**

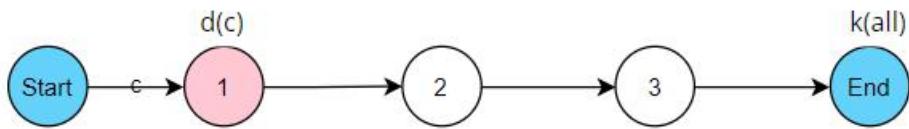


Hình 6.6.b: Đồ thị đời sống biến $b(\text{res})$ Xem thông tin chi tiết 1 sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biến $c(\text{next})$:**

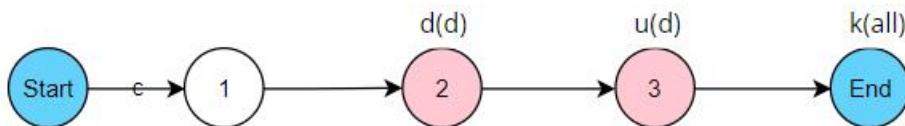


Hình 6.6.c: Đồ thị đời sóng biến $c(next)$ Xem thông tin chi tiết 1 sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biến $d(product)$:**

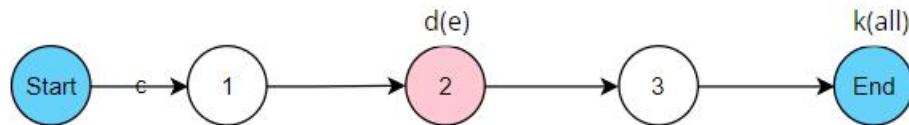


Hình 6.6.d: Đồ thị đời sóng biến $d(product)$ Xem thông tin chi tiết 1 sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biến $e(Product)$:**

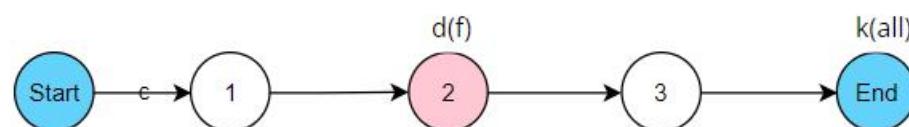


Hình 6.6.e: Đồ thị đời sóng biến $e(Product)$ Xem thông tin chi tiết 1 sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biến $f(id)$:**

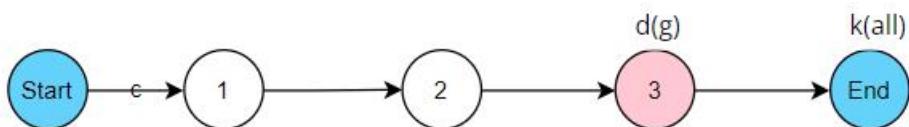


Hình 6.6.f: Đồ thị đời sóng biến f(id) Xem thông tin chi tiết 1 sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biến g(success):**



Hình 6.6.g: Đồ thị đời sóng biến g(success) Xem thông tin chi tiết 1 sản phẩm

Kịch bản 1: ~dk

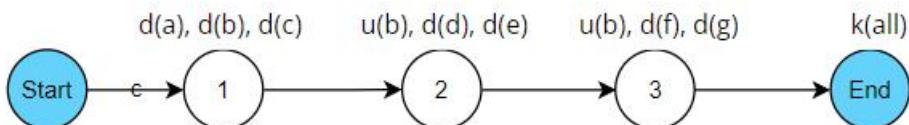
=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

10. Customer - Log out

```

//Logout User
exports.logout = catchAsyncError(async (req, res, next) => {
  res.cookie("token", null, {
    expires: new Date(Date.now()),
    httpOnly: true,
  });
  res.status(200).json({
    success: true,
    message: "Logout successfully",
  });
});
  
```

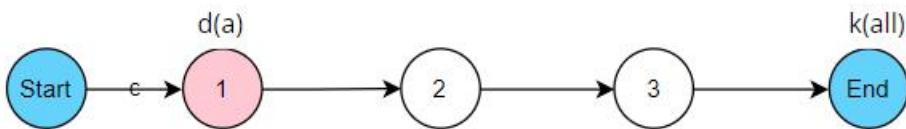
Table 6.10: Code Backend Đăng xuất tài khoản



Hình 6.10: Đồ thị dòng dữ liệu Đăng xuất tài khoản

Kiểm thử đời sóng 7 biến: a(req), b(res), c(next), d(expires), e(httpOnly), f(success), g(message)

- Kiểm thử đòi hỏi biên $a(req)$:

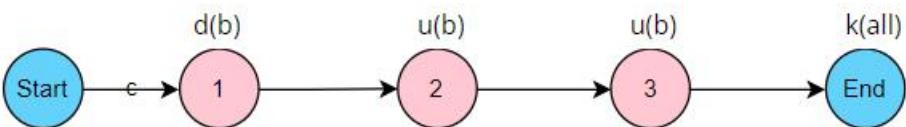


Hình 6.10.a: Đồ thị đòi hỏi biên $a(req)$ Đăng xuất tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đòi hỏi biên $b(res)$:



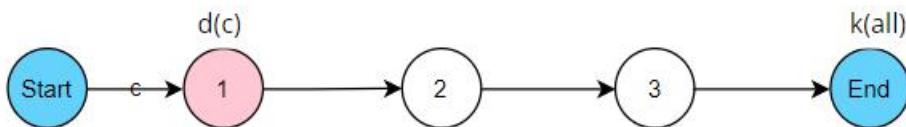
Hình 6.10.b: Đồ thị đòi hỏi biên $b(res)$ Đăng xuất tài khoản

Kịch bản 1: ~duk

Kịch bản 2: ~duuk

=> **Kết luận:** Không có cặp đôi nào của hoạt động bất thường.

- Kiểm thử đòi hỏi biên $c(next)$:

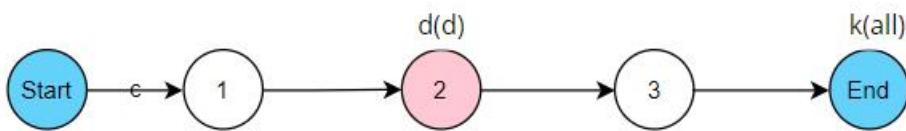


Hình 6.10.c: Đồ thị đòi hỏi biên $c(next)$ Đăng xuất tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến *d(expires)*:

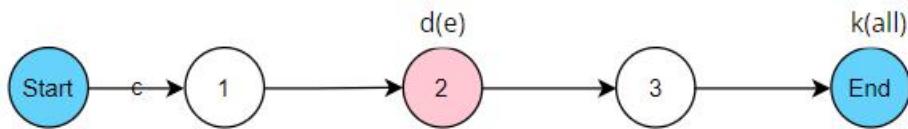


Hình 6.10.d: Đồ thị đời sống biến *d(expires)* Đăng xuất tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến *e(httpOnly)*:

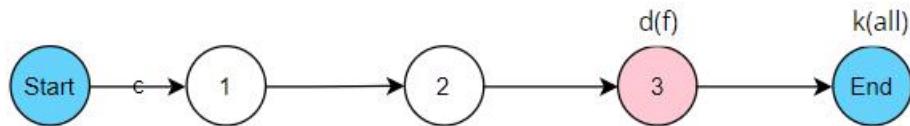


Hình 6.10.e: Đồ thị đời sống biến *e(httpOnly)* Đăng xuất tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến *f(success)*:

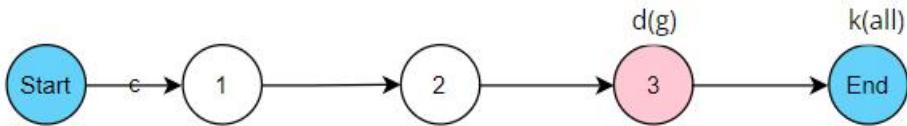


Hình 6.10.f: Đồ thị đời sống biến *f(success)* Đăng xuất tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến *g(message)*:



Hình 6.10.g: Đồ thị đời sống biến *g(message)* Đăng xuất tài khoản

Kịch bản 1: ~dk

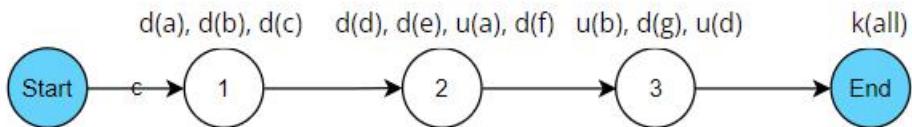
=> Kết luận: Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

11. Customer - See Profile Details

```

// Get User Detail
exports.getUserDetails = catchAsyncError(async (req, res, next) => {
  const user = await User.findById(req.user.id); /*(2)*/
  res.status(200).json({
    success: true,
    user,
  }); /*(3)*/
});
  
```

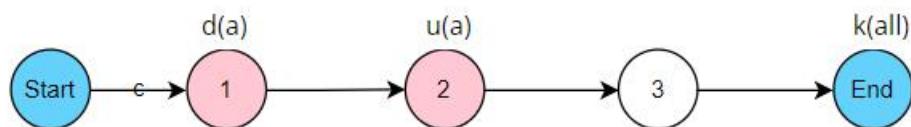
Table 6.11: Code Backend Xem thông tin chi tiết tài khoản



Hình 6.11: Đồ thị dòng dữ liệu Xem thông tin chi tiết tài khoản

Kiểm thử đời sống 7 biến: *a(req), b(res), c(next), d(user), e(User), f(id), g(success)*

- Kiểm thử đời sống biến *a(req)*:

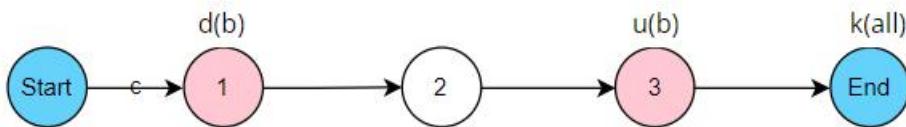


Hình 6.11.a: Đồ thị đời sống biến *a(req)* Xem thông tin chi tiết tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển b(res):**

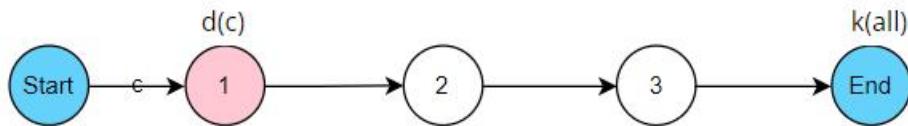


Hình 6.11.b: Đồ thị đời sống biển b(res) Xem thông tin chi tiết tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển c(next):**

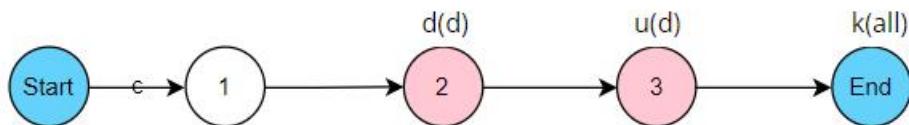


Hình 6.11.c: Đồ thị đời sống biển c(next) Xem thông tin chi tiết tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển d(user):**

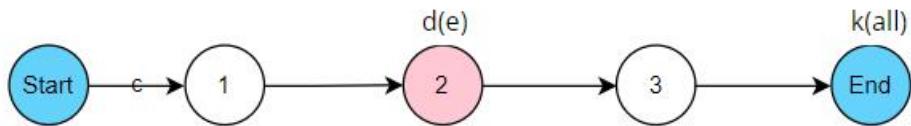


Hình 6.11.d: Đồ thị đời sống biển d(user) Xem thông tin chi tiết tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biên $e(User)$:

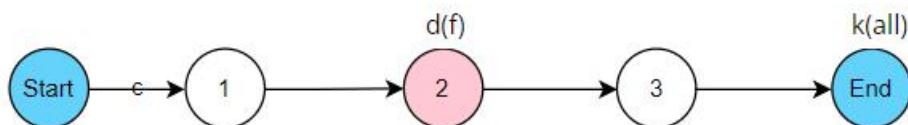


Hình 6.11.e: Đồ thị đời sóng biên $e(User)$ Xem thông tin chi tiết tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biên $f(id)$:

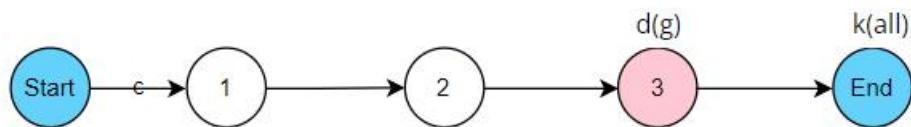


Hình 6.11.f: Đồ thị đời sóng biên $f(id)$ Xem thông tin chi tiết tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biên $g(success)$:



Hình 6.11.g: Đồ thị đời sóng biên $g(success)$ Xem thông tin chi tiết tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

12. Customer - Update Profile

```
// Update User Profile
exports.updateProfile = catchAsyncError(async (req, res, next /*(1)**/) => {
  const newData = {
    name: req.body.name,
```

```

        email: req.body.email,
    }; /*(2)*/
}

//cloudinary:
if (req.body.avatar !== "") /*(3)*/ {
    const user = await User.findById(req.user.id); /*(4)*/

    const imgId = user.avatar.public_id; /*(5)*/ //lấy avatar hiện tại
    await cloudinary.v2.uploader.destroy(imgId); /*(6)*/ //xóa avatar hiện tại

    const myCloud = await cloudinary.v2.uploader.upload(req.body.avatar, {
        folder: "avatars",
        width: 150,
        crop: "scale",
    }); /*(7)*/ //tải avatar mới lên cloudinary

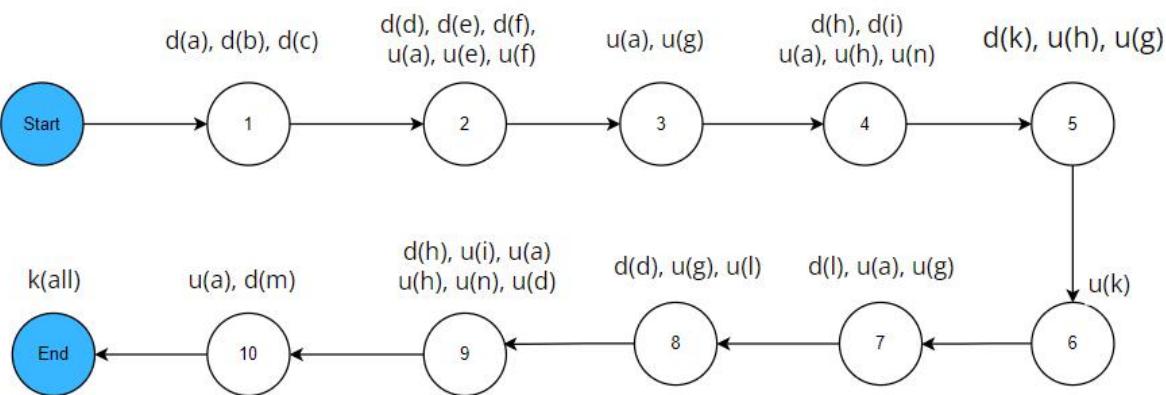
    newUserData.avatar = {
        public_id: myCloud.public_id,
        url: myCloud.secure_url,
    }; /*(8)*/
}

const user = await User.findByIdAndUpdate(req.user.id, newUserData, {
    new: true,
    runValidators: true,
    useFindAndModify: false,
}); /*(9)*/

res.status(200).json({
    success: true,
}); /*(10)*/
}

```

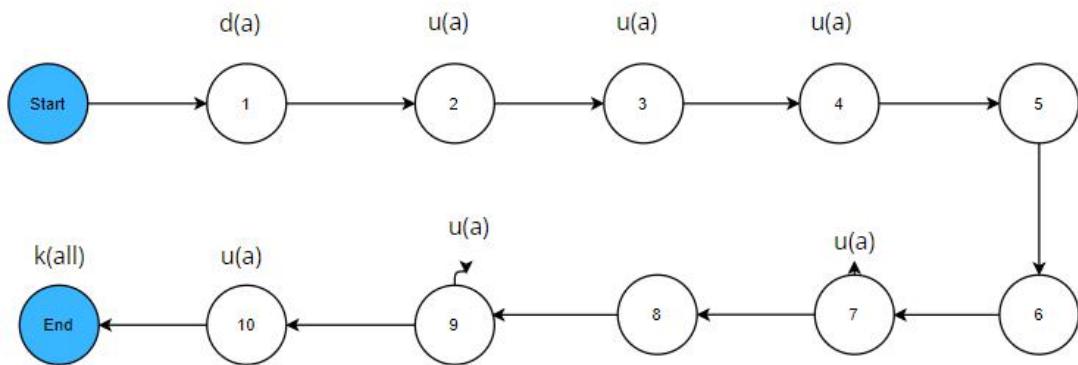
Table 6.12: Code Backend Cập nhật thông tin tài khoản



Hình 6.12: Đồ thị dòng dữ liệu Cập nhật thông tin tài khoản

Kiểm thử đời sống 12 biến: $a(req)$, $b(res)$, $c(next)$, $d(newUserData)$, $e(name)$, $f(email)$, $g/avatar$, $h(user)$, $i(User)$, $k(imageId)$, $l(myCloud)$, $m(success)$, $n(id)$

- Kiểm thử đời sống biến $a(req)$:

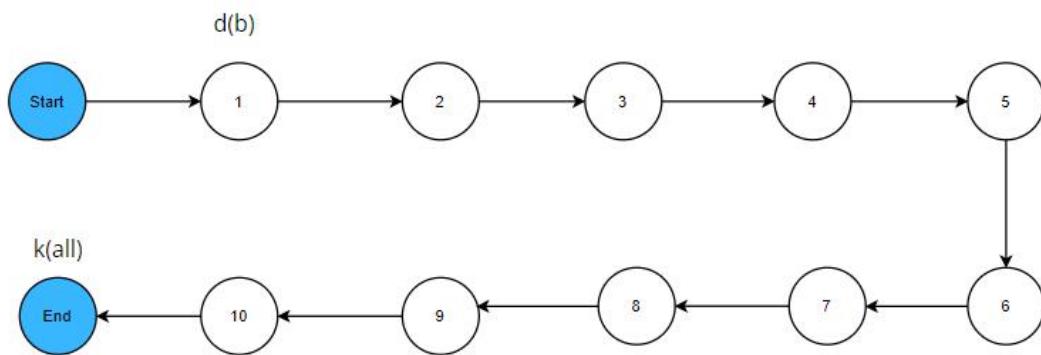


Hình 6.12.a: Đồ thị đời sống biến $a(req)$ Cập nhật thông tin tài khoản

Kịch bản 1: ~duuuuuuk

=> Kết luận: Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến $b(res)$:

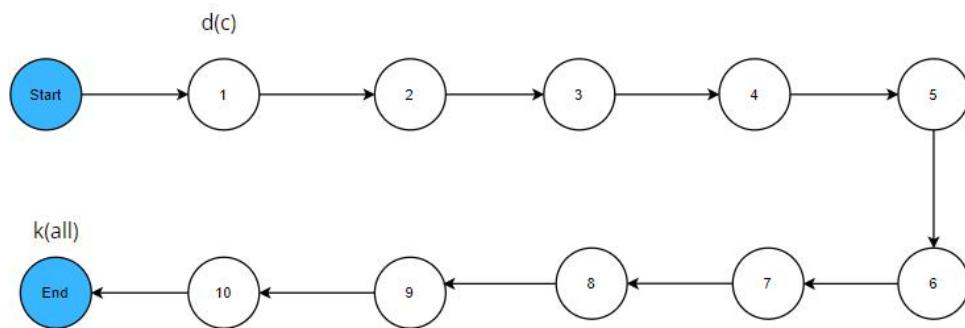


Hình 6.12.b: Đồ thị đời sống biến $b(res)$ Cập nhật thông tin tài khoản

Kịch bản 1: ~dk

=> Kết luận: Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biến $c(next)$:

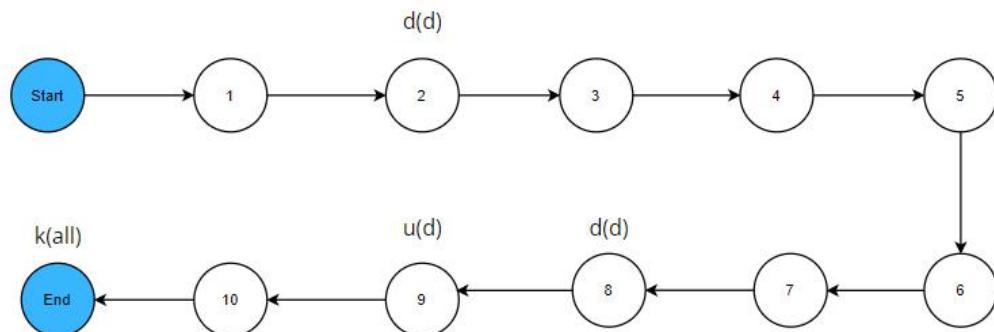


Hình 6.12.c: Đồ thị đời sóng biến $c(next)$ Cập nhật thông tin tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biến $d(newUserData)$:

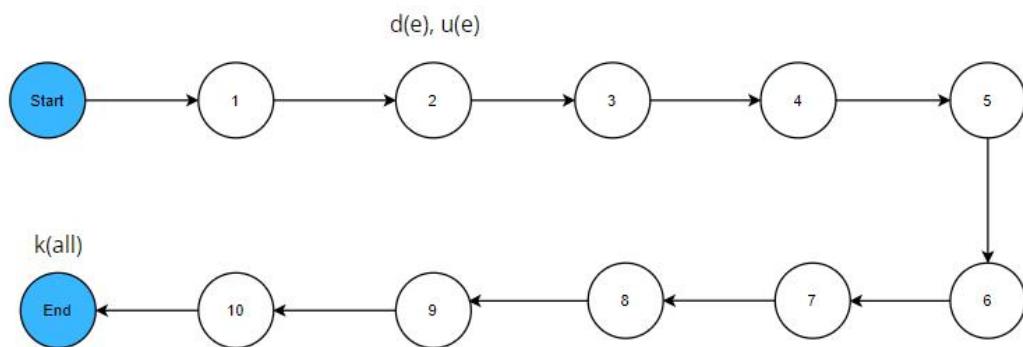


Hình 6.12.d: Đồ thị đời sóng biến $d(newUserData)$ Cập nhật thông tin tài khoản

Kịch bản 1: ~dduk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển $e(name)$:

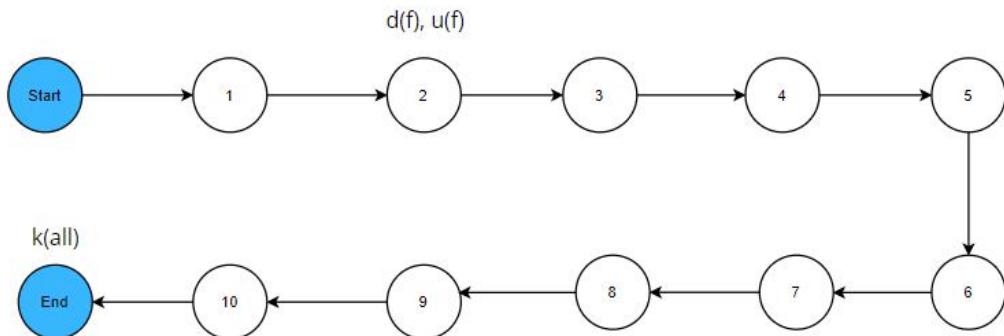


Hình 6.12.e: Đồ thị đời sóng biển $e(name)$ Cập nhật thông tin tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển $f(email)$:

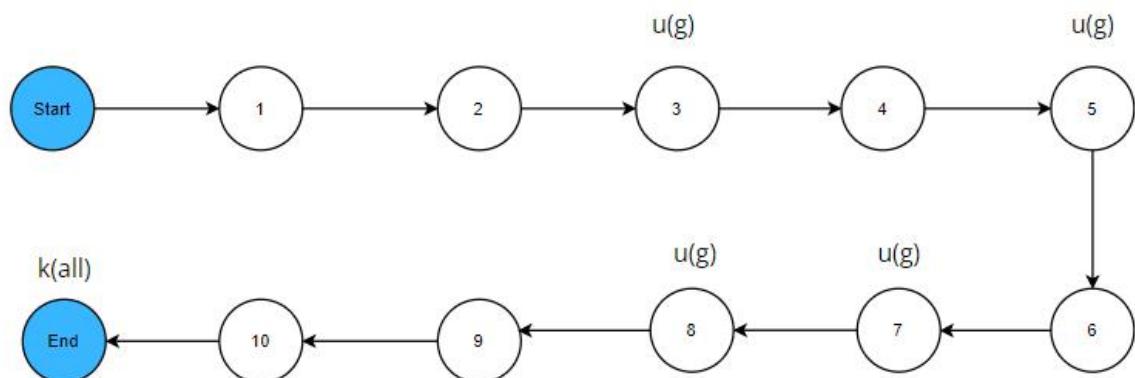


Hình 6.12.f: Đồ thị đời sóng biển $f(email)$ Cập nhật thông tin tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển g/avatar):

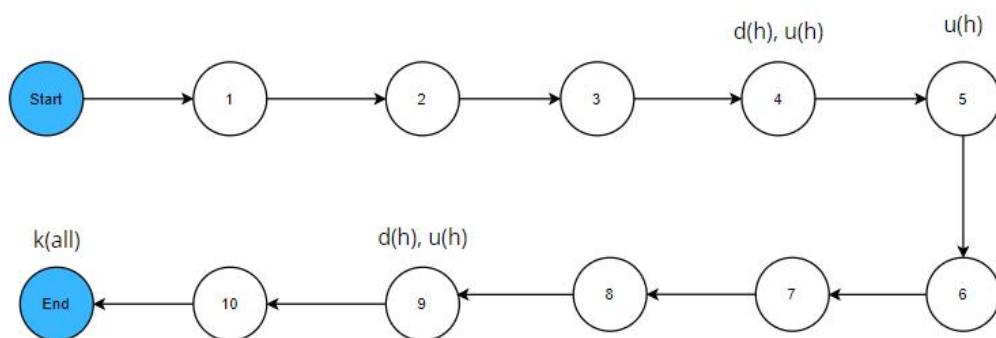


Hình 6.12.g: Đồ thị đời sóng biển g/avatar) Cập nhật thông tin tài khoản

Kịch bản 1: ~uuuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển h(user):

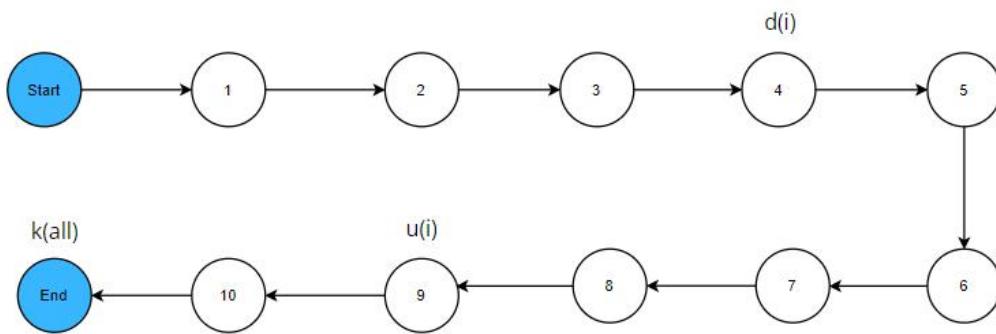


Hình 6.12.h: Đồ thị đời sóng biển h(user) Cập nhật thông tin tài khoản

Kịch bản 1: ~duuudk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển i (User):

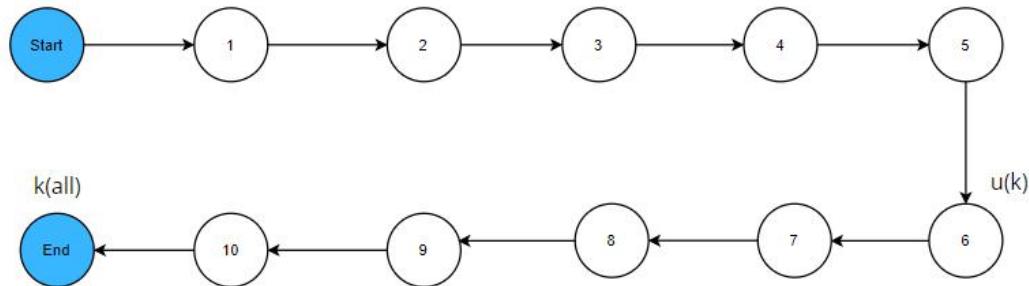


Hình 6.12.i: Đồ thị đời sóng biển i (User) Cập nhật thông tin tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển k (imageId):

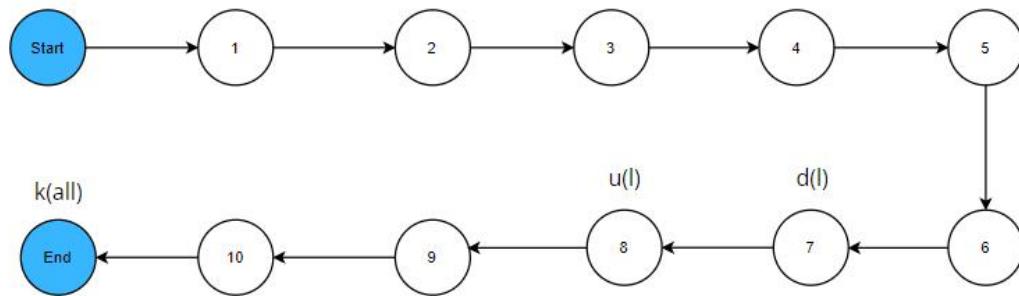


Hình 6.12.k: Đồ thị đời sóng biển k (imageId) Cập nhật thông tin tài khoản

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển l(myCloud):

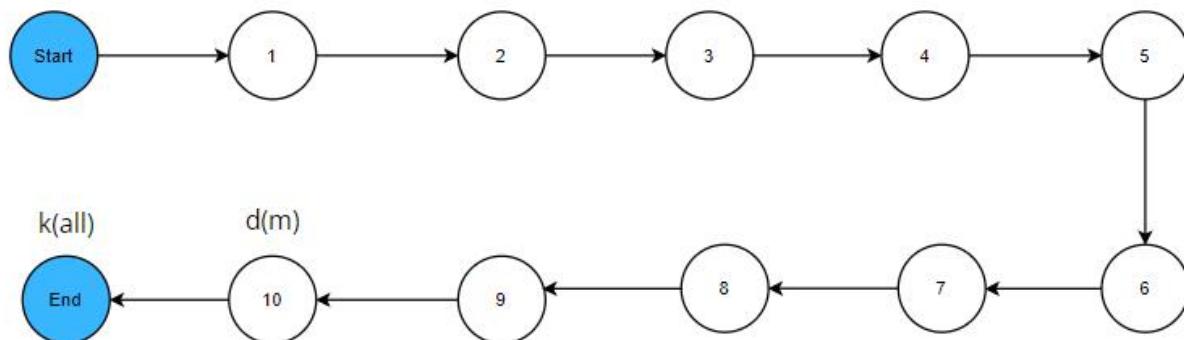


Hình 6.12.l: Đồ thị đời sóng biển l(myCloud) Cập nhật thông tin tài khoản

Kịch bản 1: ~duk

=> Kết luận: Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển m(success):



Hình 6.12.m: Đồ thị đời sóng biển m(success) Cập nhật thông tin tài khoản

Kịch bản 1: ~dk

=> Kết luận: Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

13. Customer - Update Password

```
// Update User password
exports.updatePassword = catchAsyncError(async (req, res, next) => {
  const user = await User.findById(req.user.id).select("+password"); /*(2)*/
  
```

```

const isPasswordMatched = await user.comparePassword(req.body.oldPassword); /*(3)*/

if (!isPasswordMatched /*(4)*/) {
    return next(new ErrorHander("Old password is incorrect", 400)); /*(5)*/
}

if (req.body.newPassword !== req.body.confirmPassword /*(6)*/) {
    return next(new ErrorHander("password does not match", 400)); /*(7)*/
}

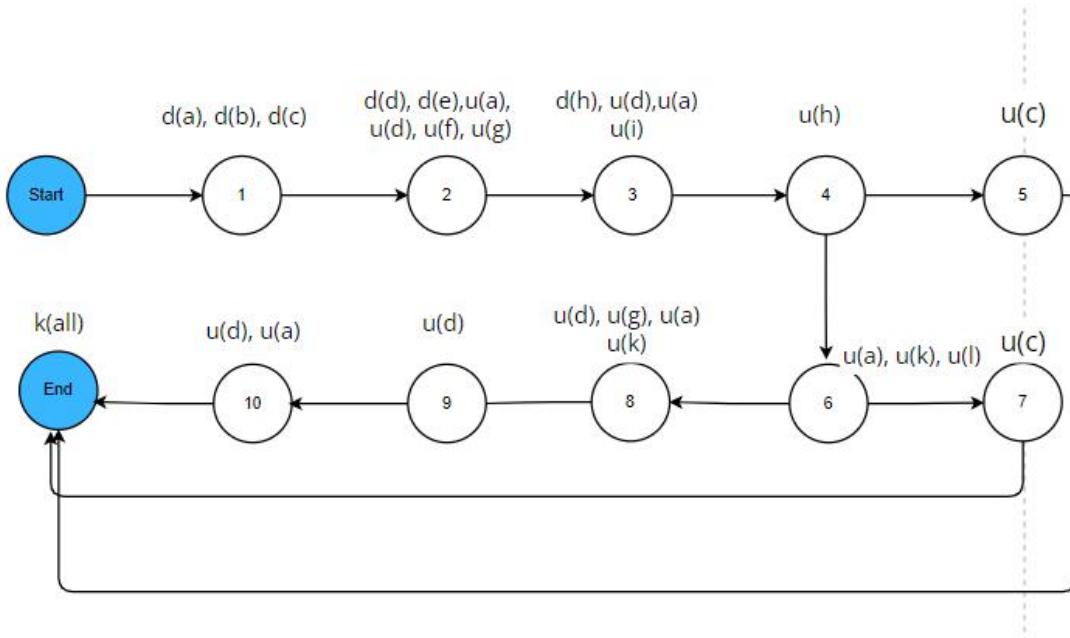
user.password = req.body.newPassword; /*(8)*/

await user.save(); /*(9)*/

sendToken(user, 200, res); /*(10)*/
);

```

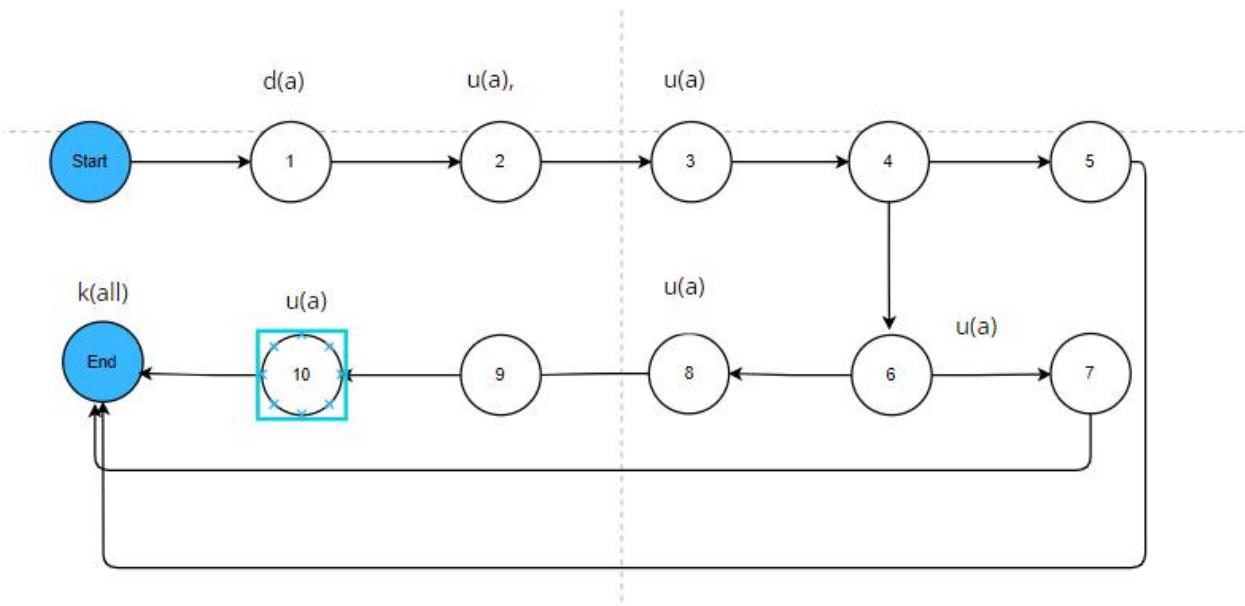
Table 6.13: Code Backend Cập nhật mật khẩu tài khoản



Hình 6.13: Đồ thị dòng dữ liệu Cập nhật mật khẩu tài khoản

Kiểm thử đời sống 11 biến: a(req), b(res), c(next), d(user), e(User), f(id), g(password), h(isPasswordMatched), i(oldPassword), k(newPassword), l(confirmPassword)

- Kiểm thử đòi hỏi biên $a(\text{req})$:



Hình 6.13.a: Đồ thị đòi hỏi biên $a(\text{req})$ Cập nhật mật khẩu tài khoản

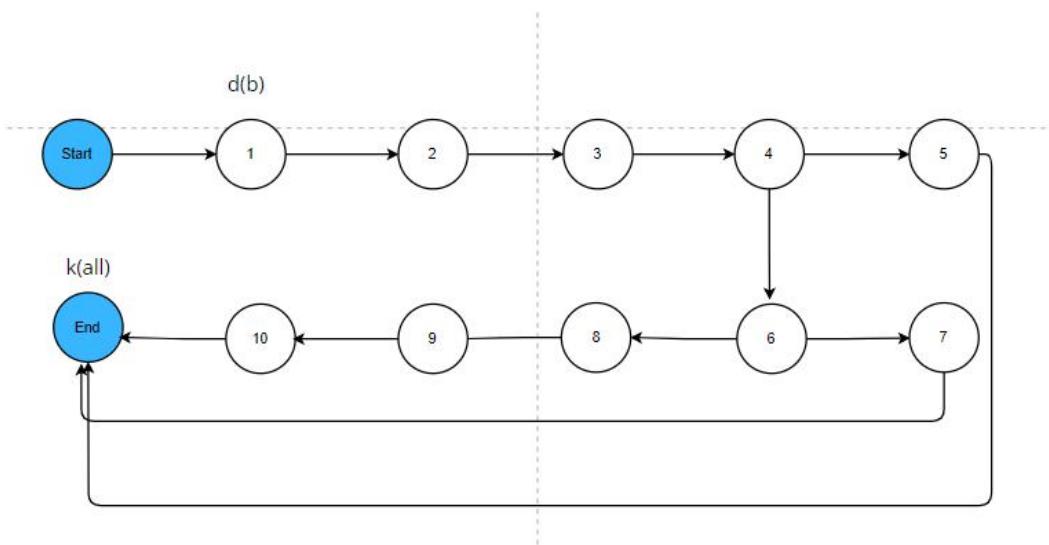
Kịch bản 1: ~duuuuuuk

Kịch bản 2: ~duuuk

Kịch bản 3: ~duuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đòi hỏi biên $b(\text{req})$:



Hình 6.13.a: Đồ thị đòi hỏi biên $b(\text{req})$ Cập nhật mật khẩu tài khoản

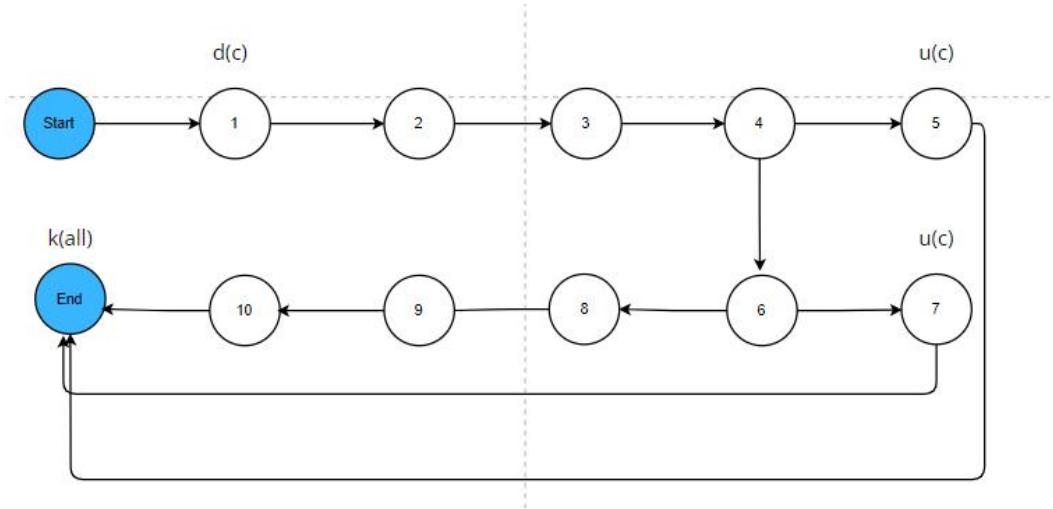
Kịch bản 1: ~dk

Kịch bản 2: ~dk

Kịch bản 3: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- **Kiểm thử đòn tay c(next):**



Hình 6.13.a: Đồ thị đòn tay c(next) Cập nhật mật khẩu tài khoản

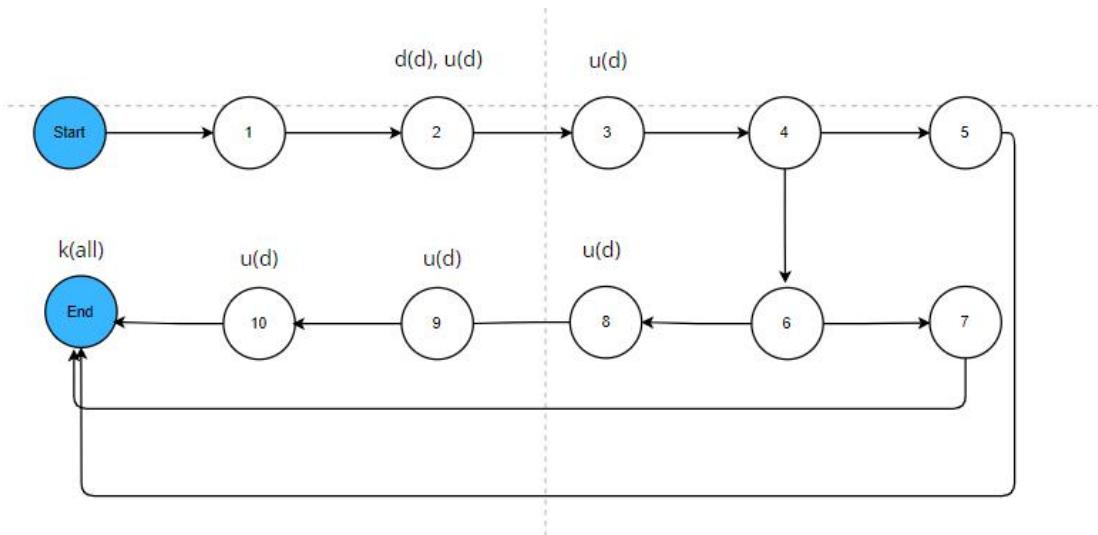
Kịch bản 1: ~dk

Kịch bản 2: ~duk

Kịch bản 3: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển $d(user)$:



Hình 6.13.a: Đồ thị đời sóng biển $d(user)$ Cập nhật mật khẩu tài khoản

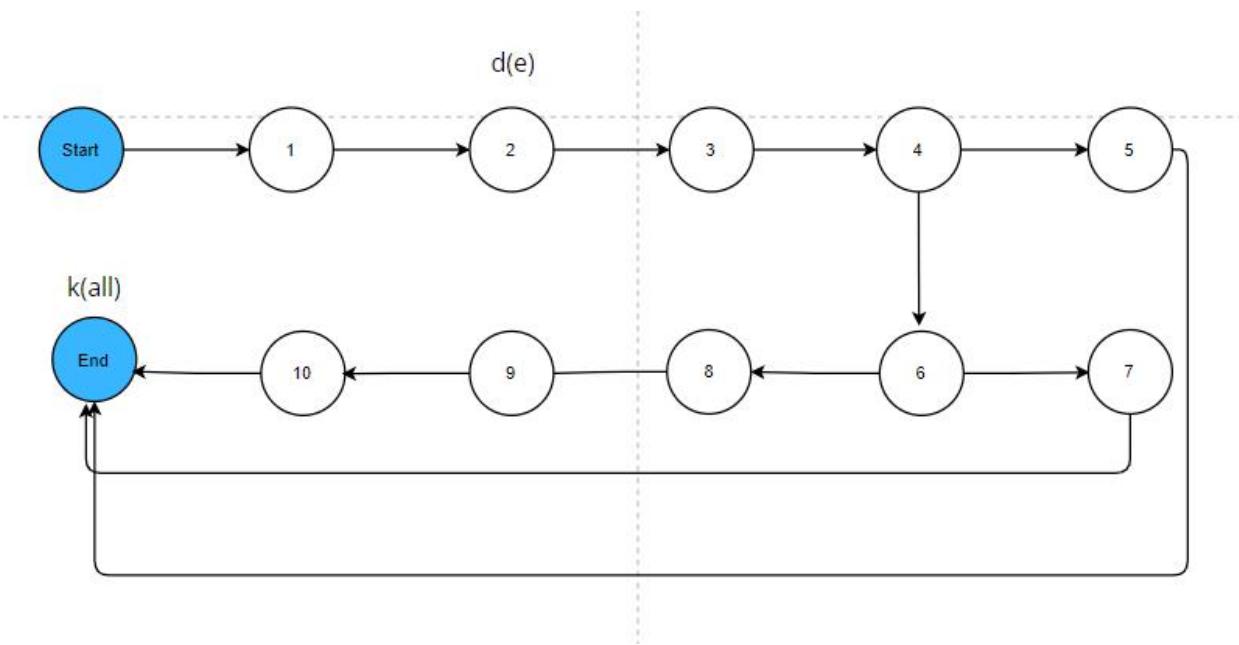
Kịch bản 1: ~duuuuuuk

Kịch bản 2: ~duuuk

Kịch bản 3: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sóng biển $e(User)$:



Hình 6.13.a: Đồ thị đời sóng biển $e(User)$ Cập nhật mật khẩu tài khoản

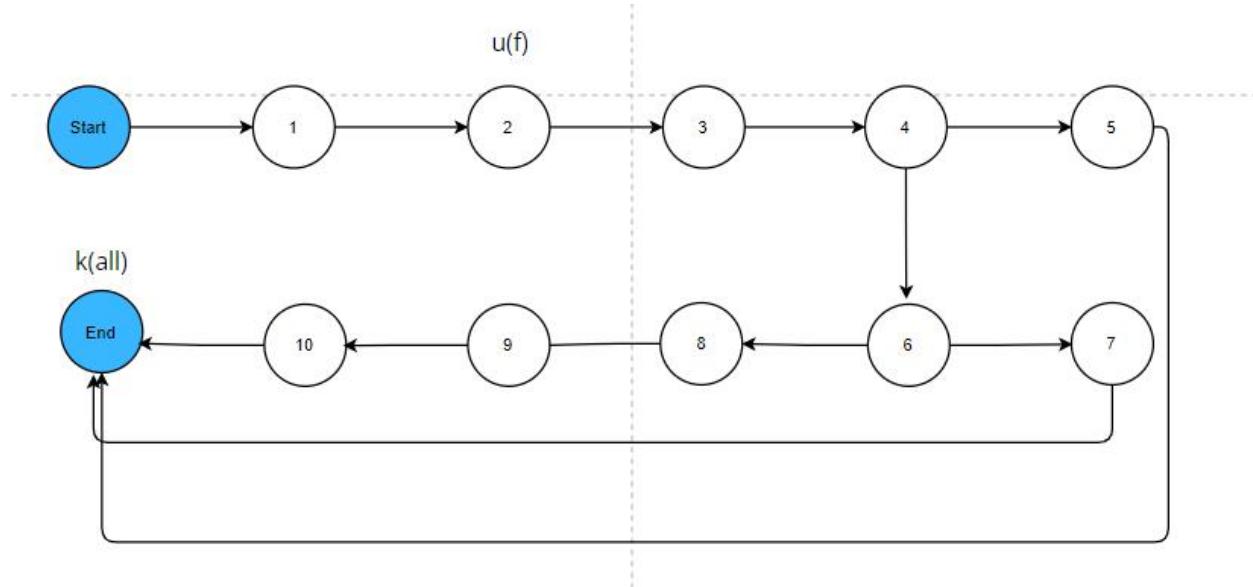
Kịch bản 1: ~dk

Kịch bản 2: ~dk

Kịch bản 3: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- **Kiểm thử đời sống biển f(id):**



Hình 6.13.a: Đồ thị đời sống biển f(id) Cập nhật mật khẩu tài khoản

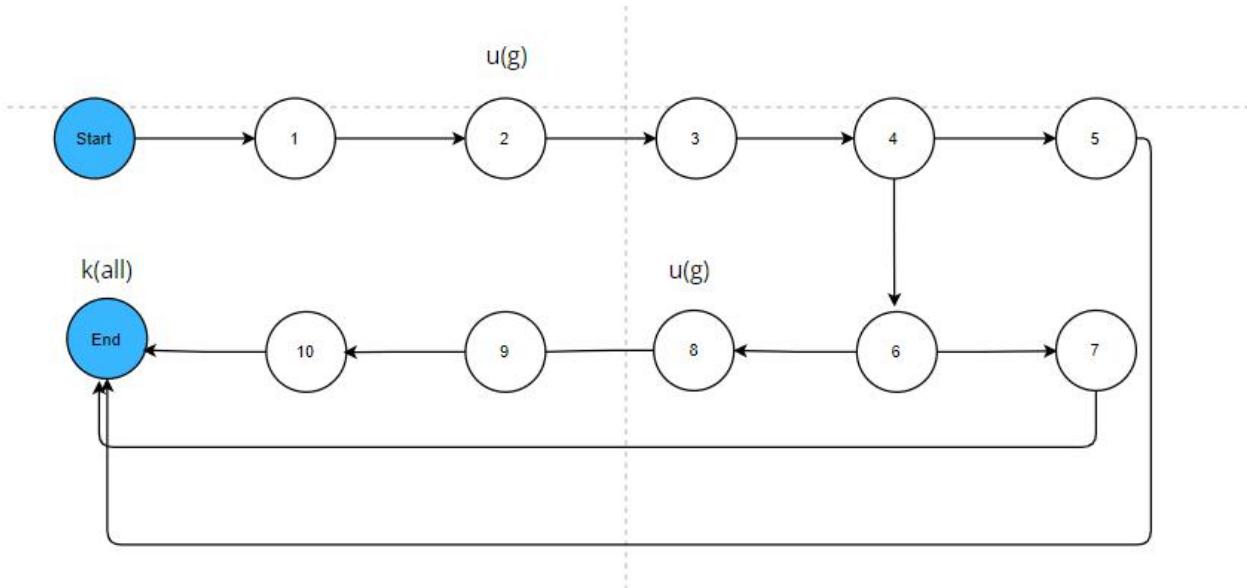
Kịch bản 1: ~uk

Kịch bản 2: ~uk

Kịch bản 3: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đòi hỏi biến g (password):



Hình 6.13.a: Đồ thị đòi hỏi biến g (password) Cập nhật mật khẩu tài khoản

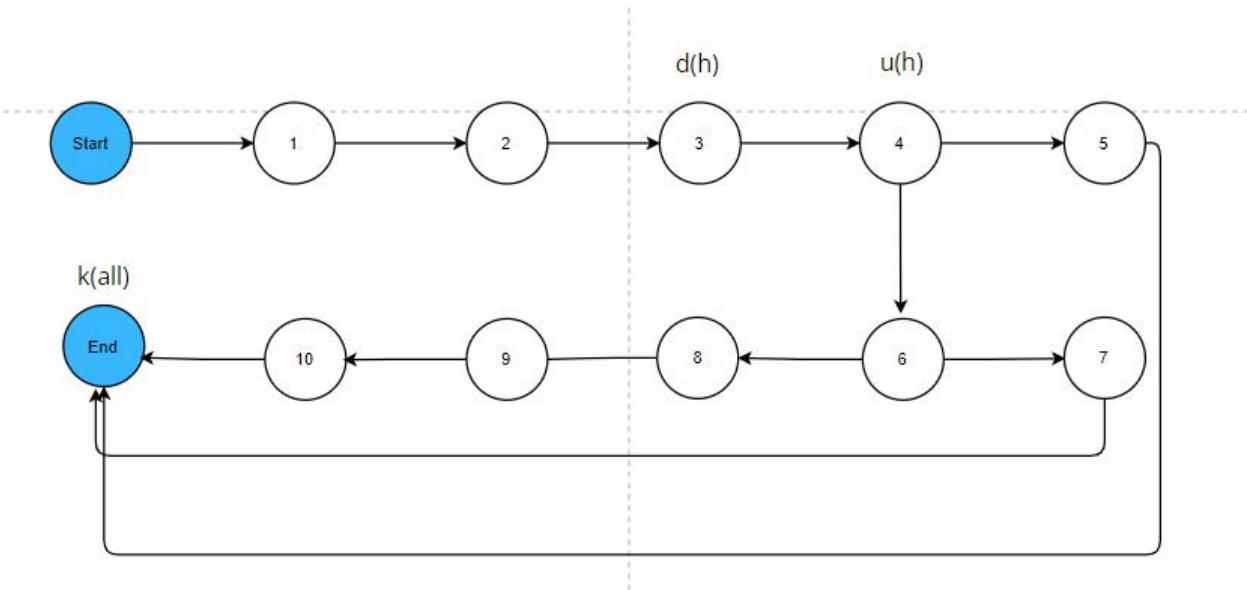
Kịch bản 1: ~uuk

Kịch bản 2: ~uk

Kịch bản 3: ~uk

=> Kết luận: Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đòi hỏi biến h (isPasswordMatched):



Hình 6.13.a: Đồ thị đổi sóng biến $h(\text{isPasswordMatched})$ Cập nhật mật khẩu tài khoản

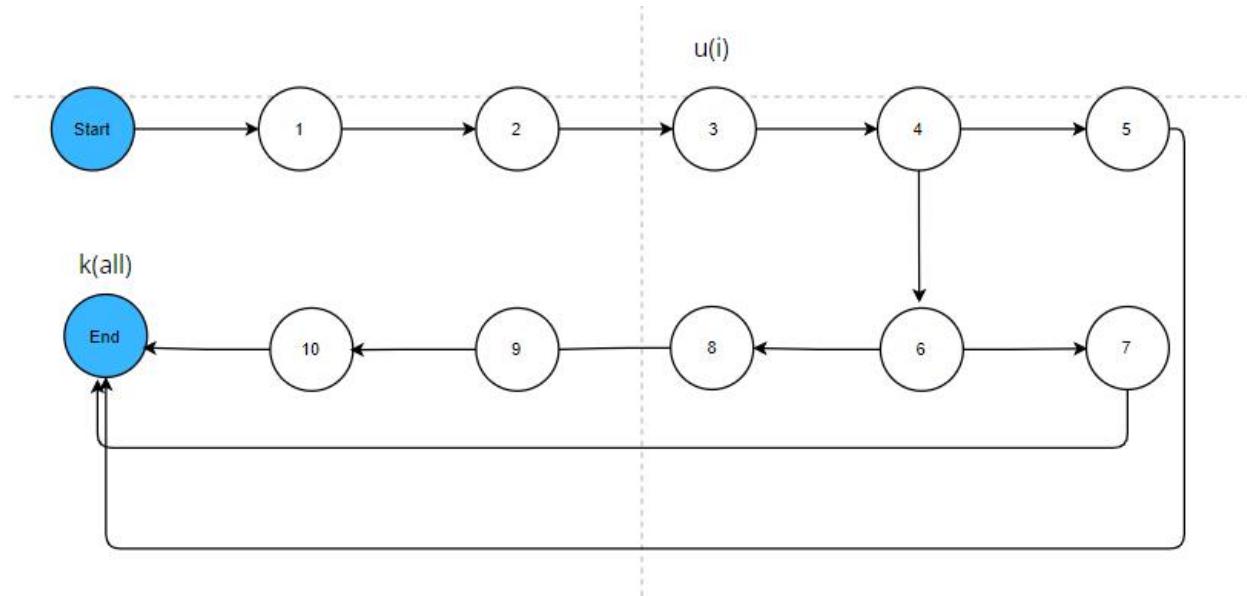
Kịch bản 1: ~duk

Kịch bản 2: ~duk

Kịch bản 3: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- **Kiểm thử đổi sóng biến $i(\text{oldPassword})$:**



Hình 6.13.a: Đồ thị đổi sóng biến $i(\text{oldPassword})$ Cập nhật mật khẩu tài khoản

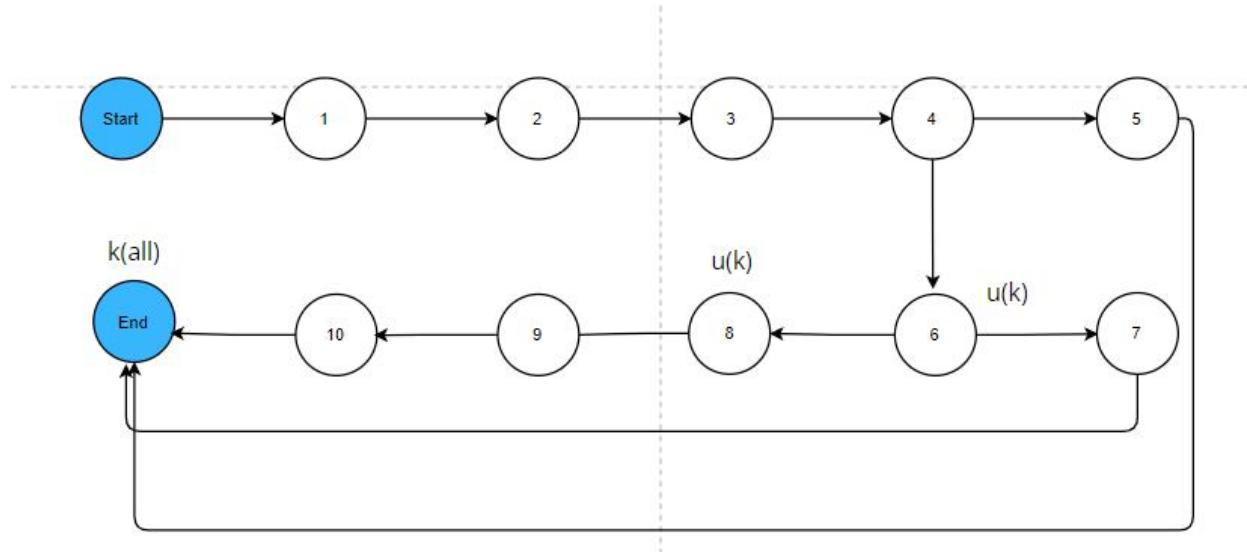
Kịch bản 1: ~uk

Kịch bản 2: ~uk

Kịch bản 3: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đổi sóng biến k (newPassword):



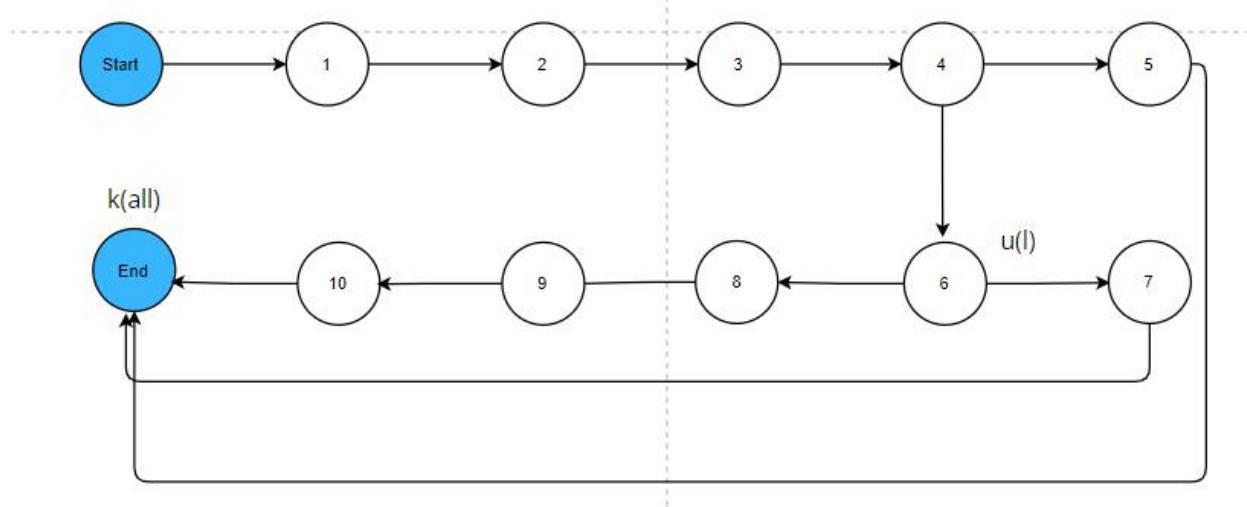
Hình 6.13.a: Đồ thị đổi sóng biến k (newPassword) Cập nhật mật khẩu tài khoản
Kịch bản 1: ~uuk

Kịch bản 2: ~k

Kịch bản 3: ~uk

=> Kết luận: Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đổi sóng biến l (confirmPassword):



Hình 6.13.a: Đồ thị đổi sóng biến l (confirmPassword) Cập nhật mật khẩu tài khoản
Kịch bản 1: ~k

Kịch bản 2: ~uk

Kịch bản 3: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

14. Customer - Create A Product Review

```
//Create A New Review or Update A Review
exports.createProductReview = catchAsyncError(async (req, res, next) => {
  const { rating, comment, productId } = req.body; /*(1)*/
  const review = {
    user: req.user._id,
    name: req.user.name,
    rating: Number(rating),
    comment,
  }; /*(3)*/
  const product = await Product.findById(productId); /*(4)*/

  const isReviewed = product.reviews.find(
    (rev) => rev.user.toString() === req.user._id.toString()
  ); /*(5)*/

  if (isReviewed /*(6)*/) {
    product.reviews.forEach((rev) => {
      if (rev.user.toString() === req.user._id.toString()) /*(7)*/
        (rev.rating = rating /*(8)*), (rev.comment = comment /*(9)*);
    });
  } else {
    product.reviews.push(review); /*(10)*/
    product.numOfReviews = product.reviews.length; /*(11)*/
  }

  let avg = 0; /*(12)*/

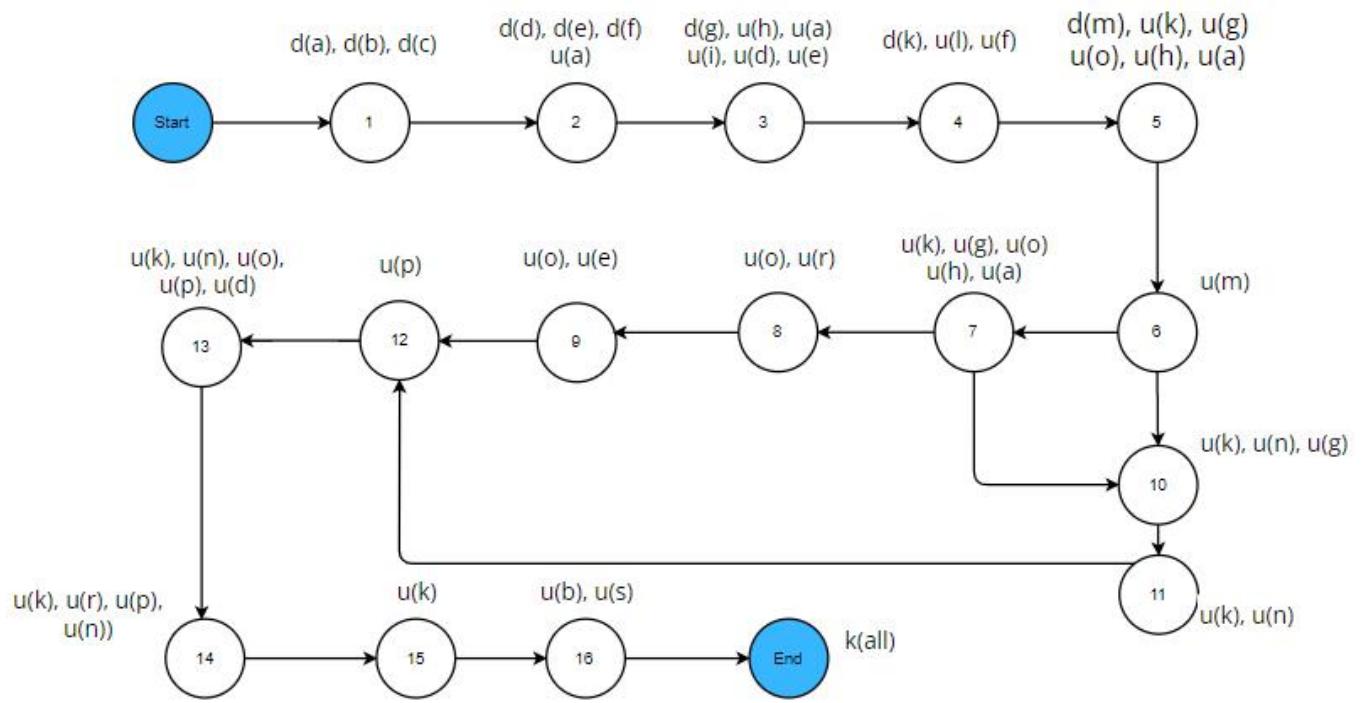
  product.reviews.forEach((rev) => {
    avg += rev.rating;
  }); /*(13)*/

  product.ratings = avg / product.reviews.length; /*(14)*/

  await product.save({ validateBeforeSave: false }); /*(15)*/

  res.status(200).json({
    success: true,
  }); /*(16)*/
});
```

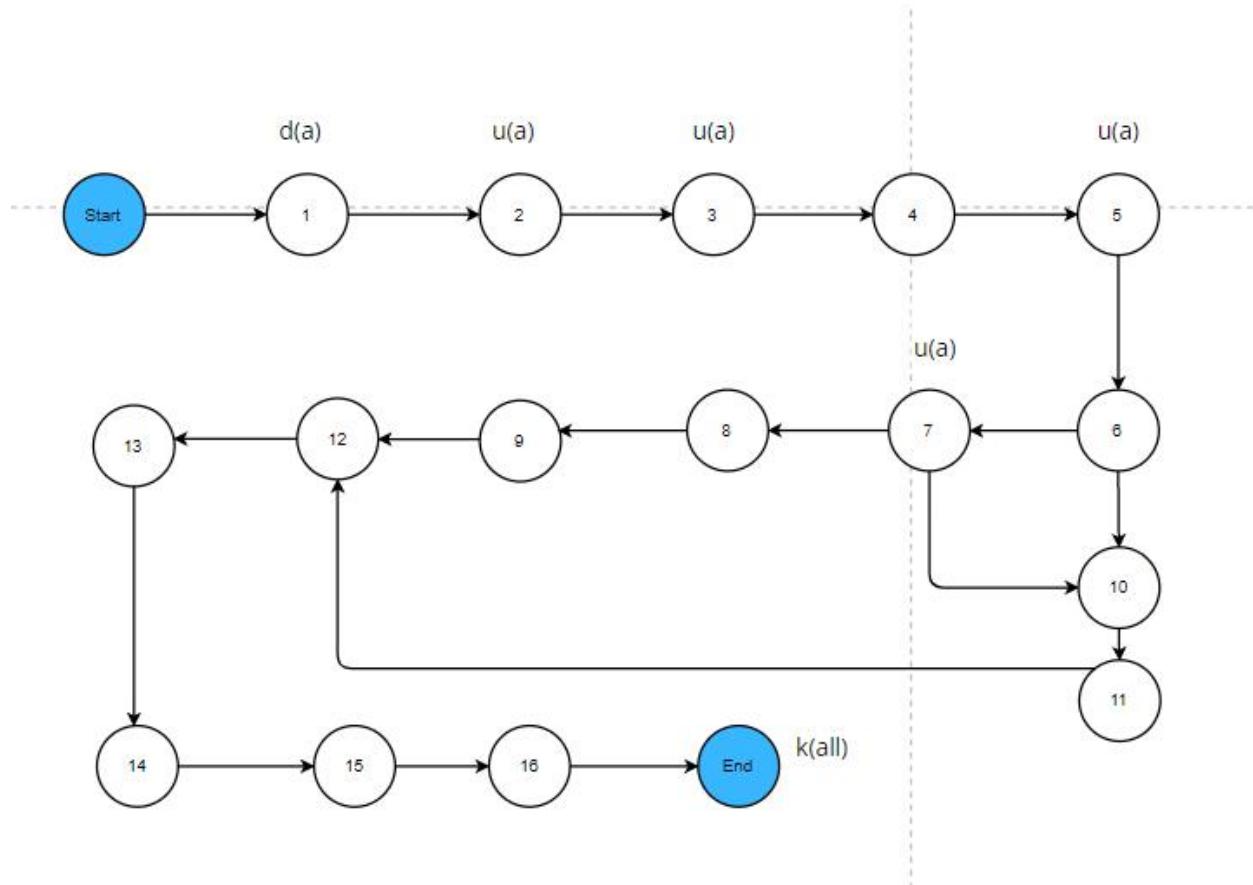
Table 6.14: Code Backend Đánh giá 1 sản phẩm



Hình 6.14: Đồ thị dòng dữ liệu Dánh giá 1 sản phẩm

Kiểm thử dòi sống 17 biến: *a(req)*, *b(res)*, *c(next)*, *d(rating)*, *e(comment)*, *f(productId)*, *g(review)*, *h(user)*, *i(name)*, *k(product)*, *l(Product)*, *m(isReviewed)*, *n(reviews)*, *o(rev)*, *p(avg)*, *r(ratings)*, *s(success)*

- Kiểm thử đời sống biển a(req):



Hình 6.14.a: Đồ thị đời sống biển a(req) Dánh giá 1 sản phẩm

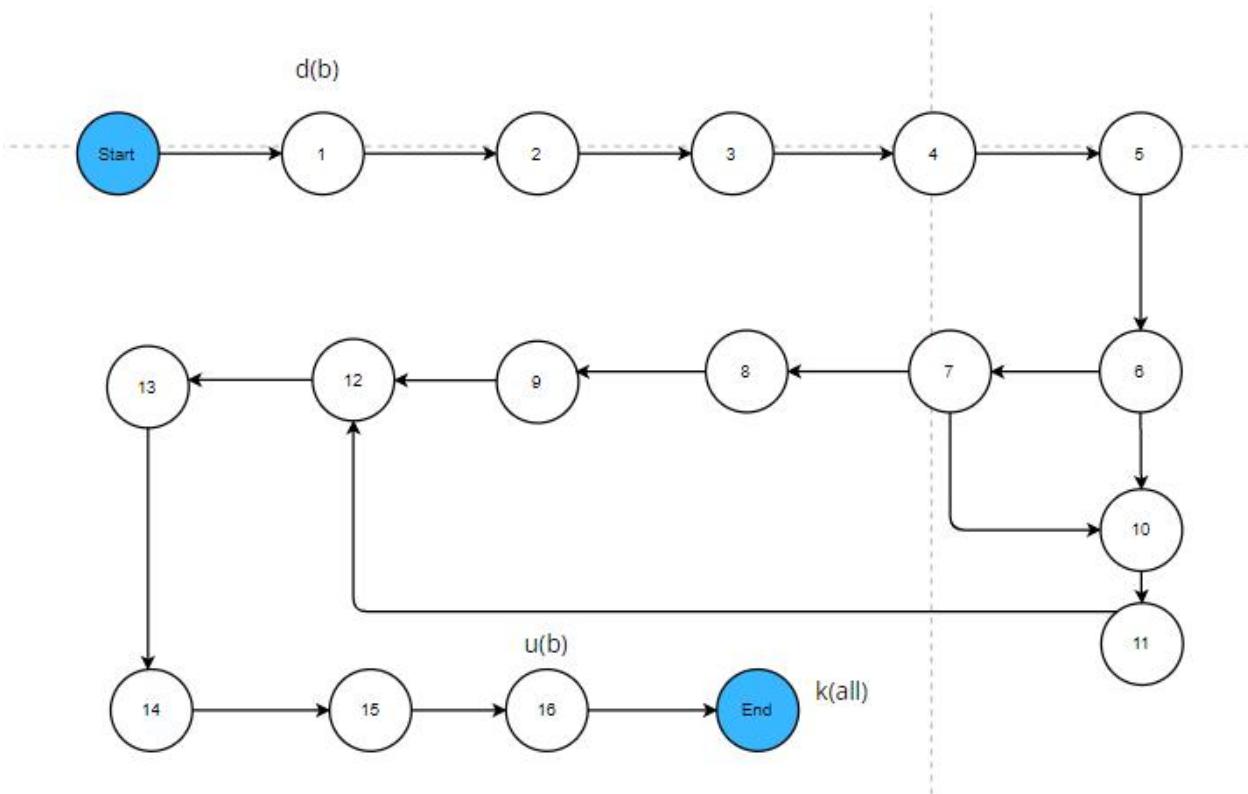
Kịch bản 1: ~duuuuk

Kịch bản 2: ~duuuk

Kịch bản 3: ~duuuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển b(res):



Hình 6.14.b: Đồ thị đời sống biển b(res) Đánh giá 1 sản phẩm

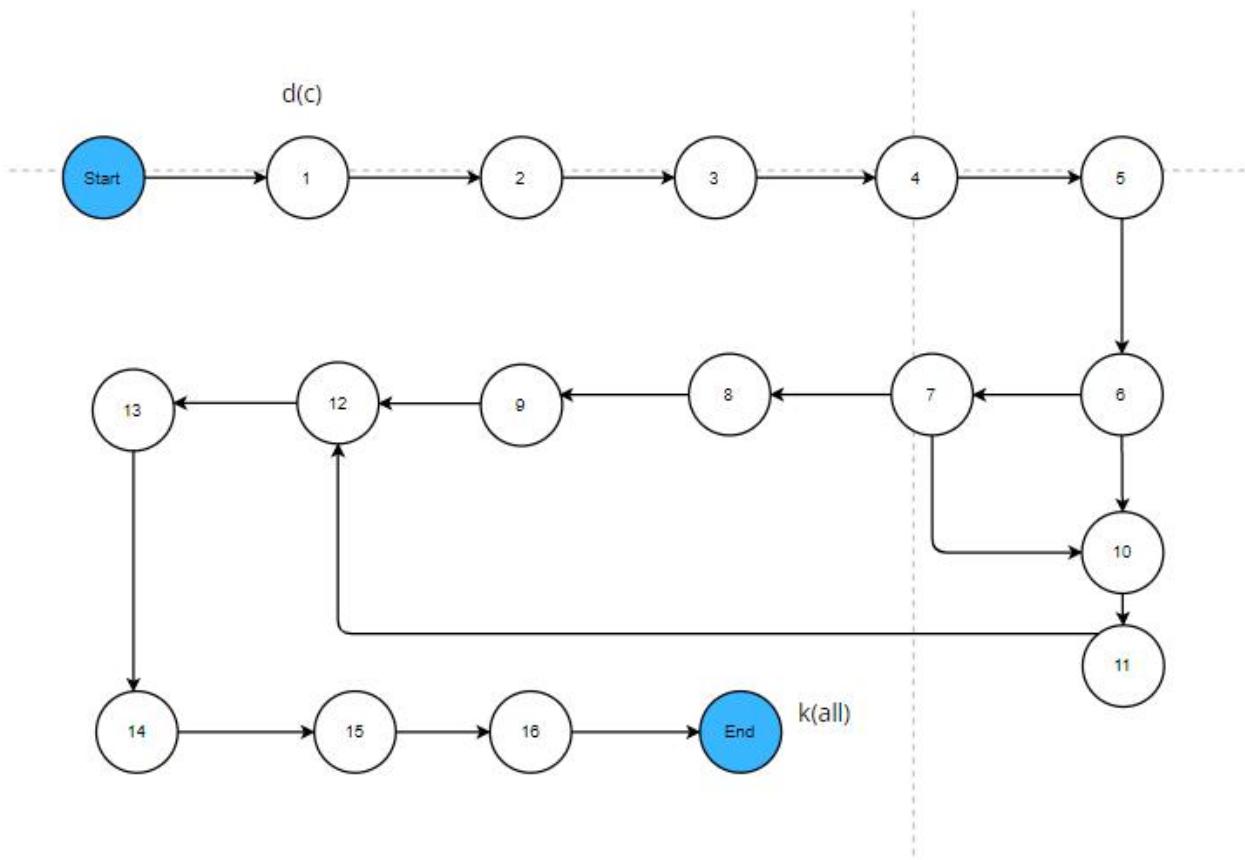
Kịch bản 1: ~duk

Kịch bản 2: ~duk

Kịch bản 3: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đòn bẩy c(next):



Hình 6.14.c: Đồ thị đòn bẩy c(next) Dánh giá 1 sản phẩm

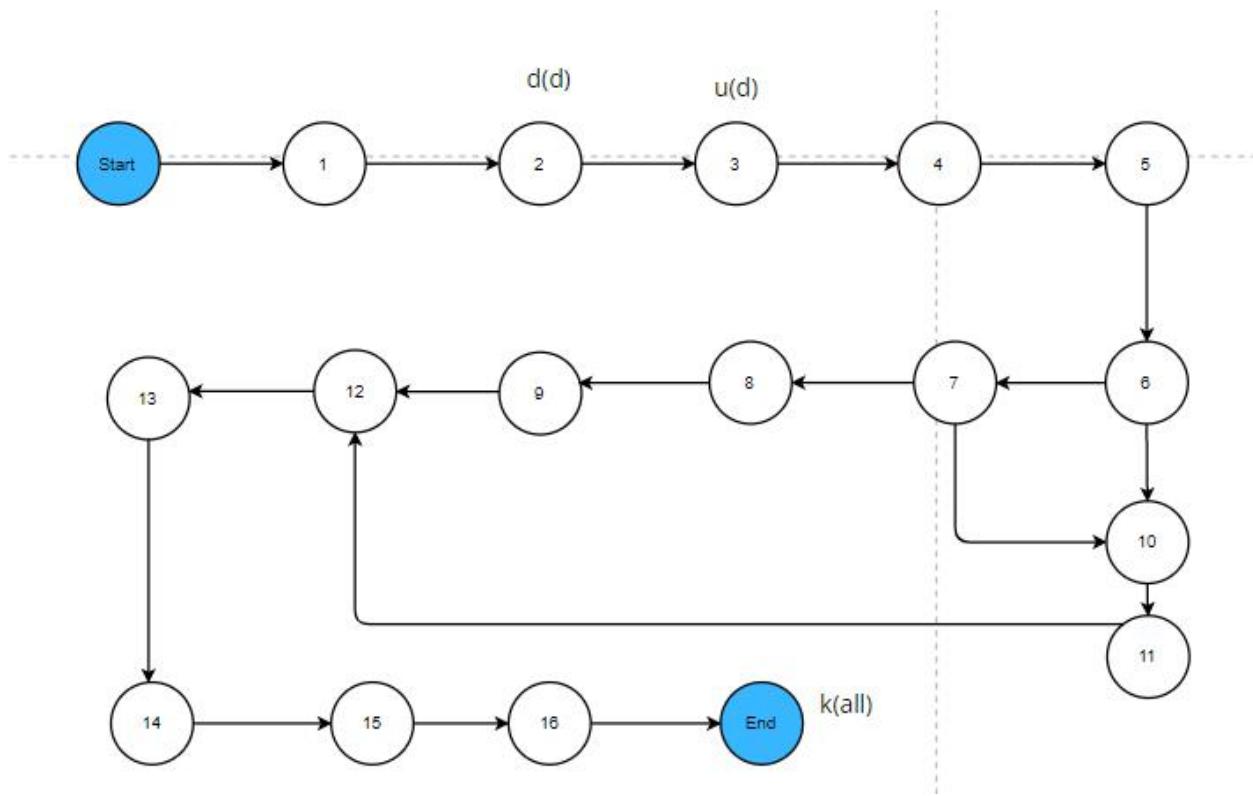
Kịch bản 1: ~dk

Kịch bản 2: ~dk

Kịch bản 3: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đòi hỏi biên d(rating):



Hình 6.14.d: Đồ thị đòi hỏi sóng biên d(rating) Đánh giá 1 sản phẩm

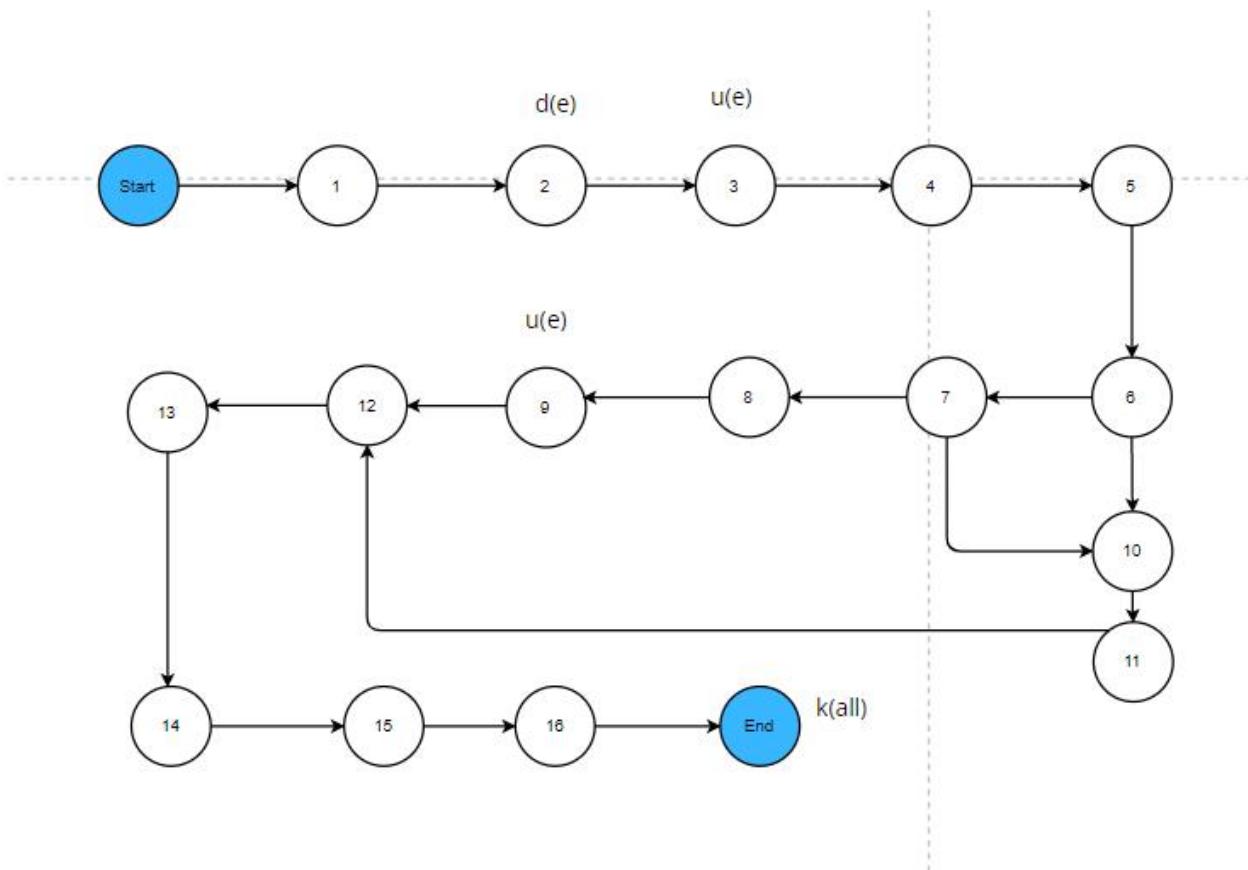
Kịch bản 1: ~duk

Kịch bản 2: ~duk

Kịch bản 3: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển e(comment):



Hình 6.14.e: Đồ thị đời sống biển e(comment) Đánh giá 1 sản phẩm

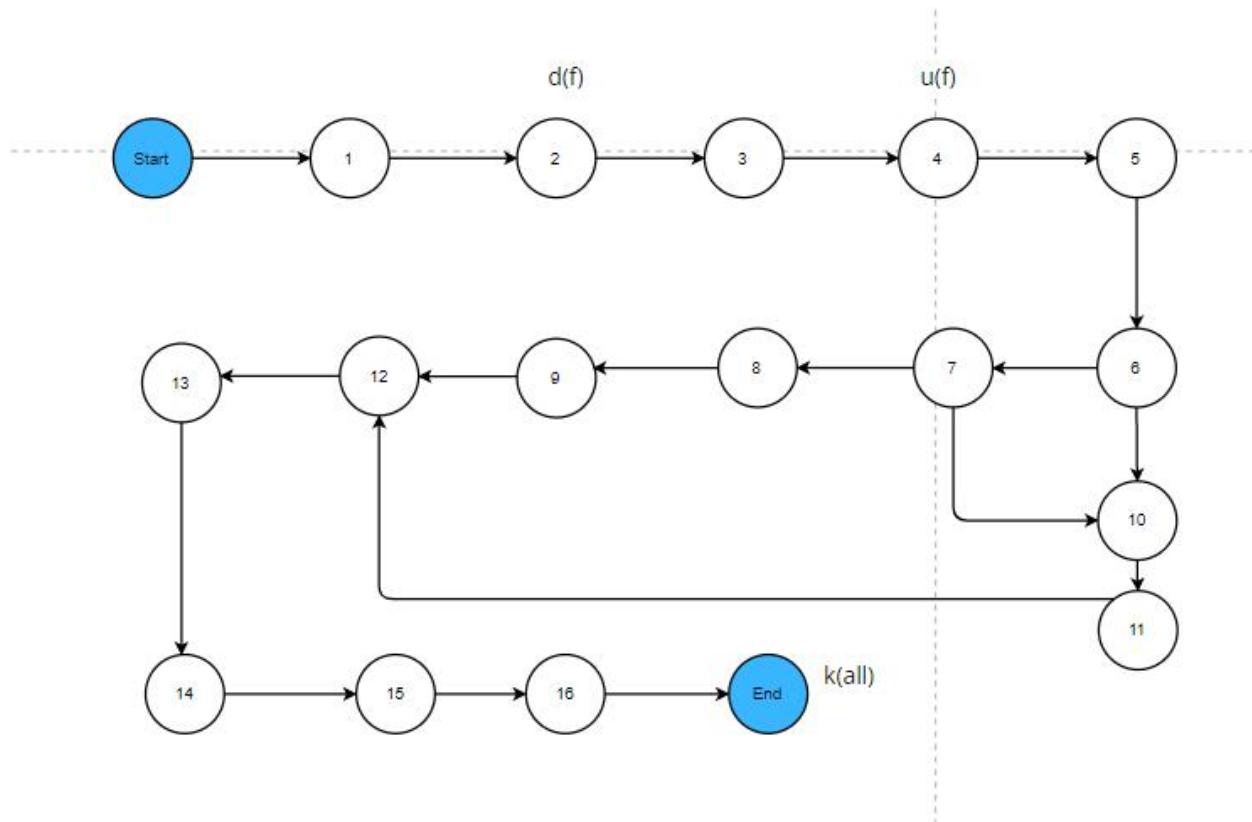
Kịch bản 1: ~duuk

Kịch bản 2: ~duk

Kịch bản 3: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biến $f(productId)$:



Hình 6.14.f: Đồ thị đời sống biến $f(productId)$ Đánh giá 1 sản phẩm

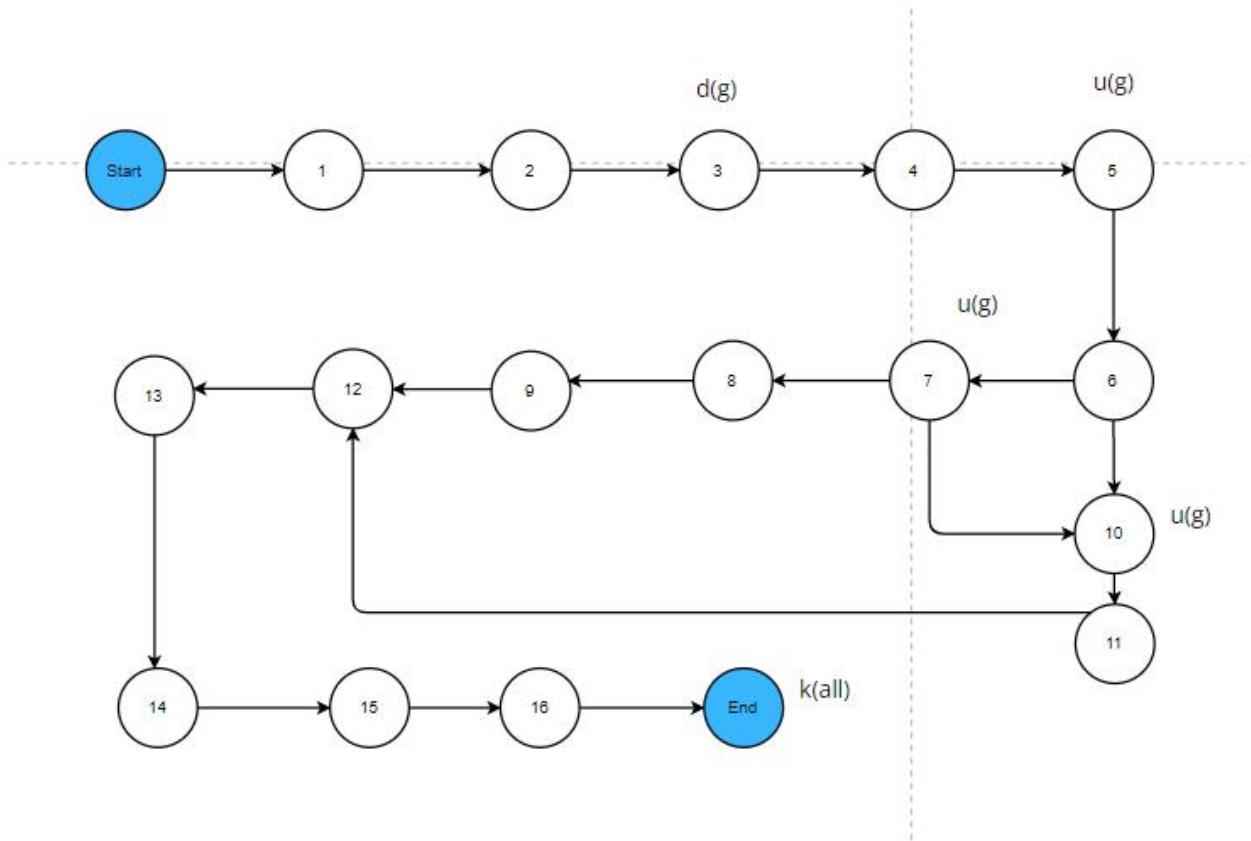
Kịch bản 1: ~duk

Kịch bản 2: ~duk

Kịch bản 3: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển g(review):



Hình 6.14.g: Đồ thị đời sống biển g(review) Đánh giá 1 sản phẩm

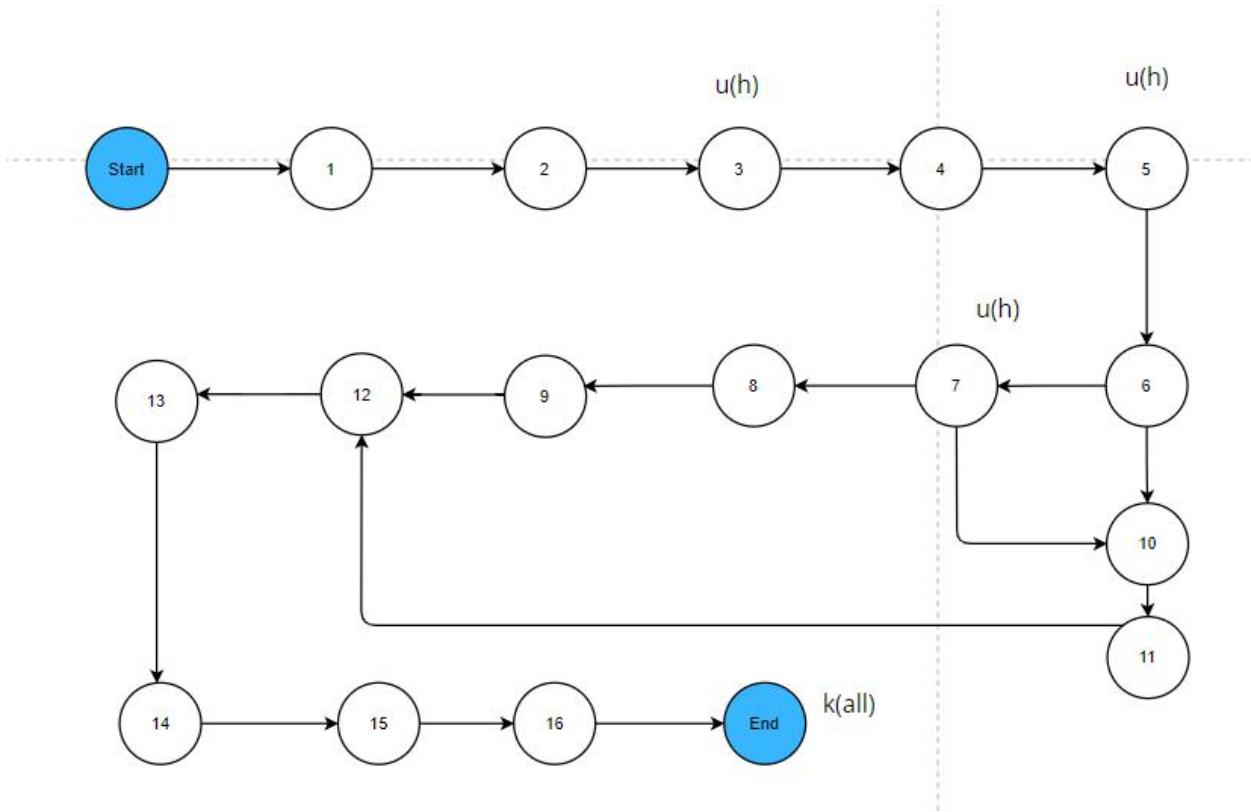
Kịch bản 1: ~duuk

Kịch bản 2: ~duuk

Kịch bản 3: ~duuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển $h(\text{user})$:



Hình 6.14.h: Đồ thị đời sống biển $h(\text{user})$ Đánh giá 1 sản phẩm

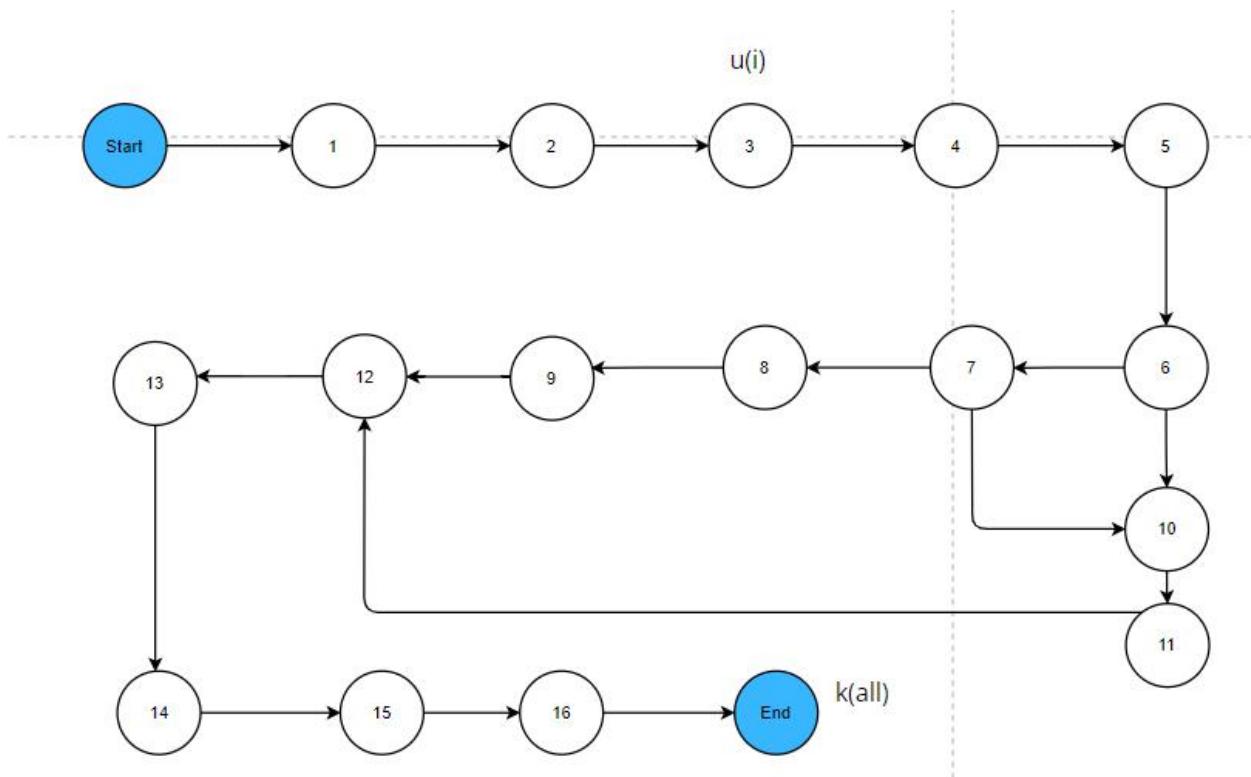
Kịch bản 1: ~uuuk

Kịch bản 2: ~uuk

Kịch bản 3: ~uuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển $i(name)$:



Hình 6.14.i: Đồ thị đời sống biển $i(name)$ Đánh giá 1 sản phẩm

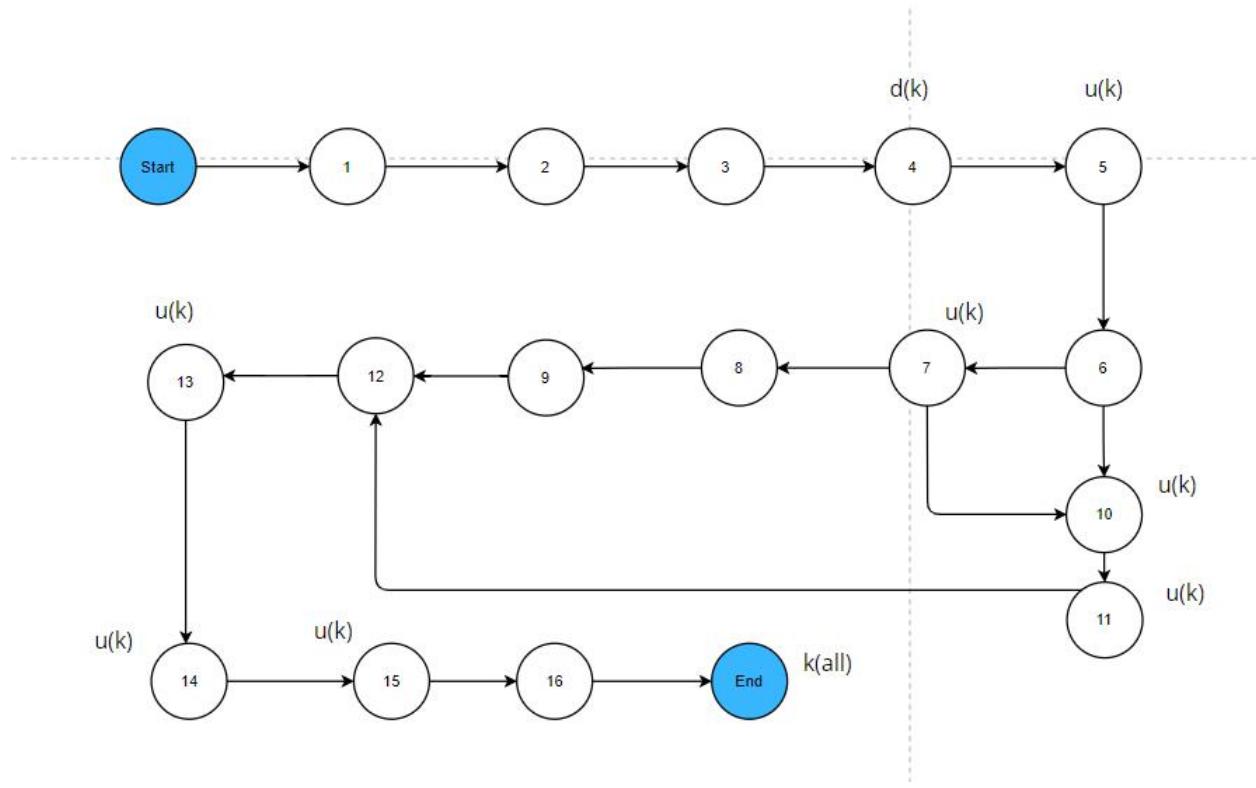
Kịch bản 1: ~uk

Kịch bản 2: ~uk

Kịch bản 3: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biến k (product):



Hình 6.14.k: Đồ thị đời sống biến k (product) Dánh giá 1 sản phẩm

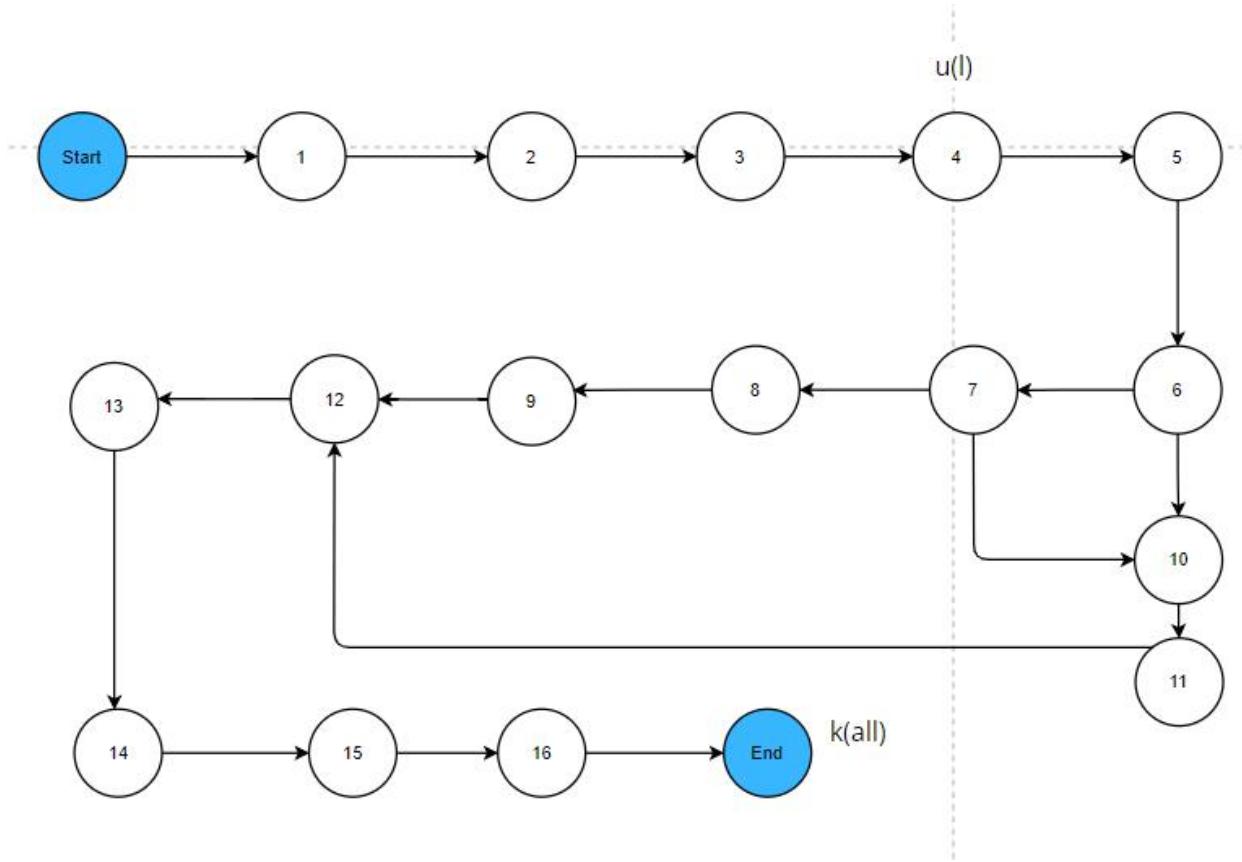
Kịch bản 1: ~duuuuuk

Kịch bản 2: ~duuuuuuk

Kịch bản 3: ~duuuuuuuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biến l(Product):



Hình 6.14.l: Đồ thị đời sống biến l(Product) Đánh giá 1 sản phẩm

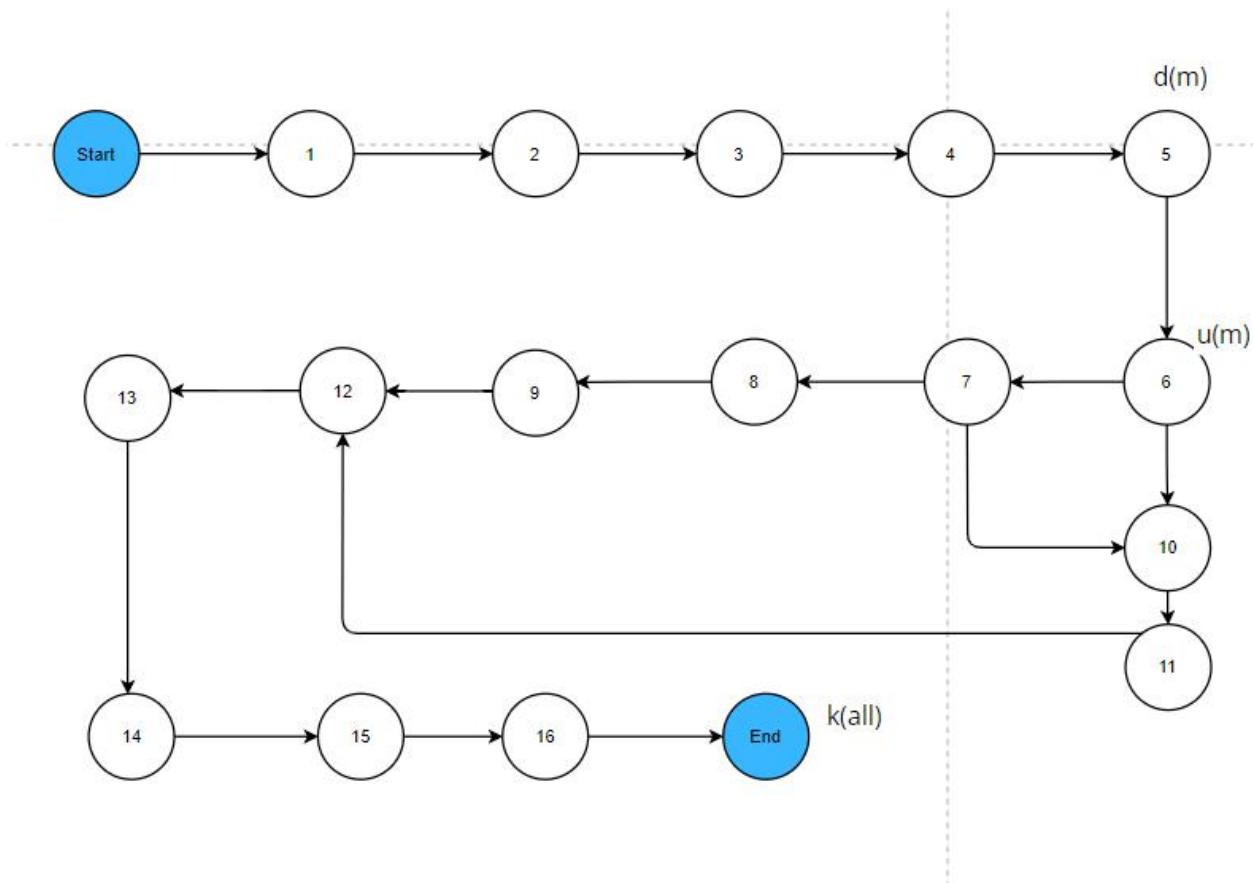
Kịch bản 1: ~uk

Kịch bản 2: ~uk

Kịch bản 3: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển m(isReviewed):



Hình 6.14.m: Đồ thị đời sống biển m(isReviewed) Đánh giá 1 sản phẩm

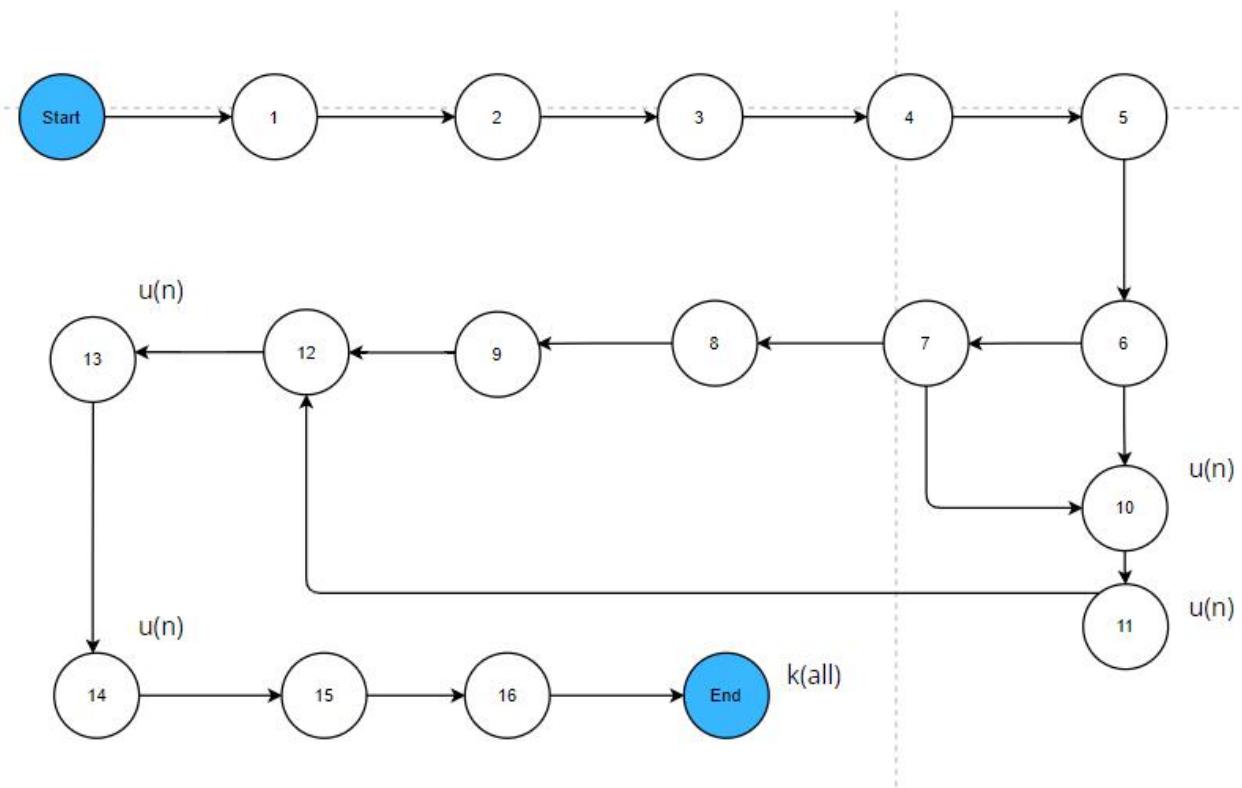
Kịch bản 1: ~duk

Kịch bản 2: ~duk

Kịch bản 3: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển n(reviews):



Hình 6.14.n: Đồ thị đời sống biển n(reviews) Đánh giá 1 sản phẩm

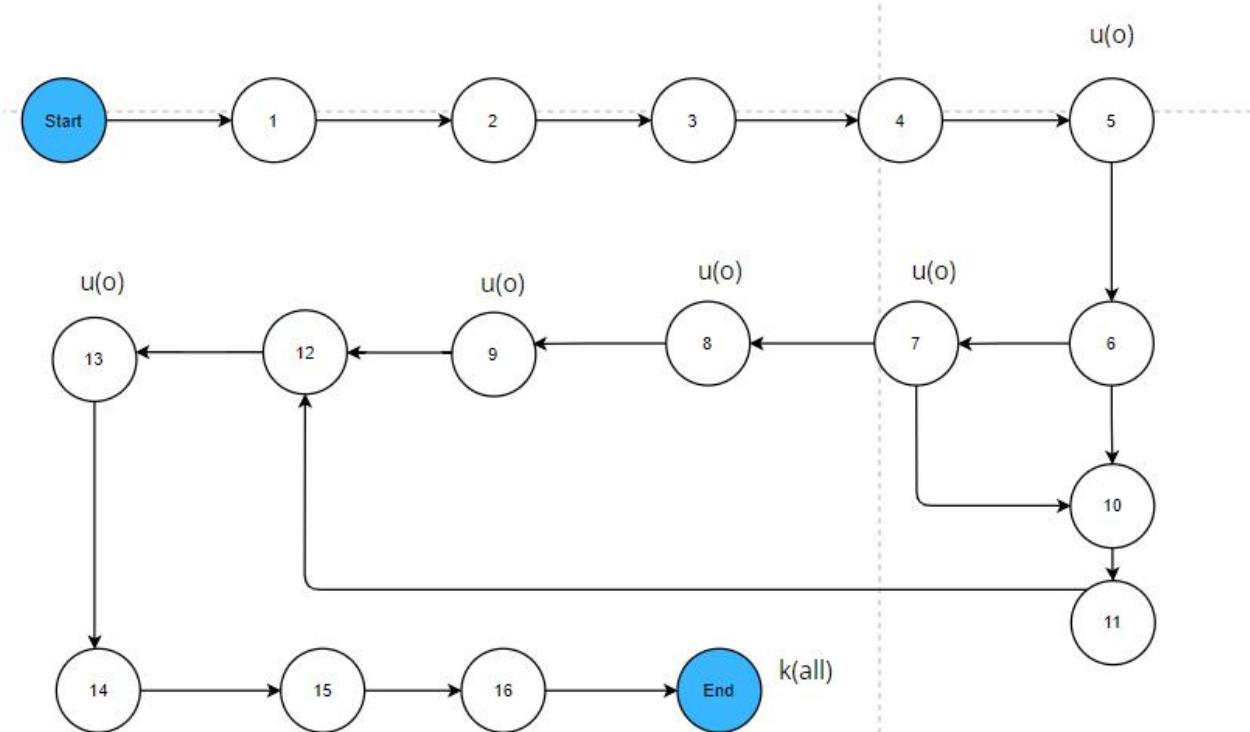
Kịch bản 1: ~uuuk

Kịch bản 2: ~uuuuuk

Kịch bản 3: ~uuuuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển $o(\text{rev})$:



Hình 6.14.o: Đồ thị đời sống biển $o(\text{rev})$ Đánh giá 1 sản phẩm

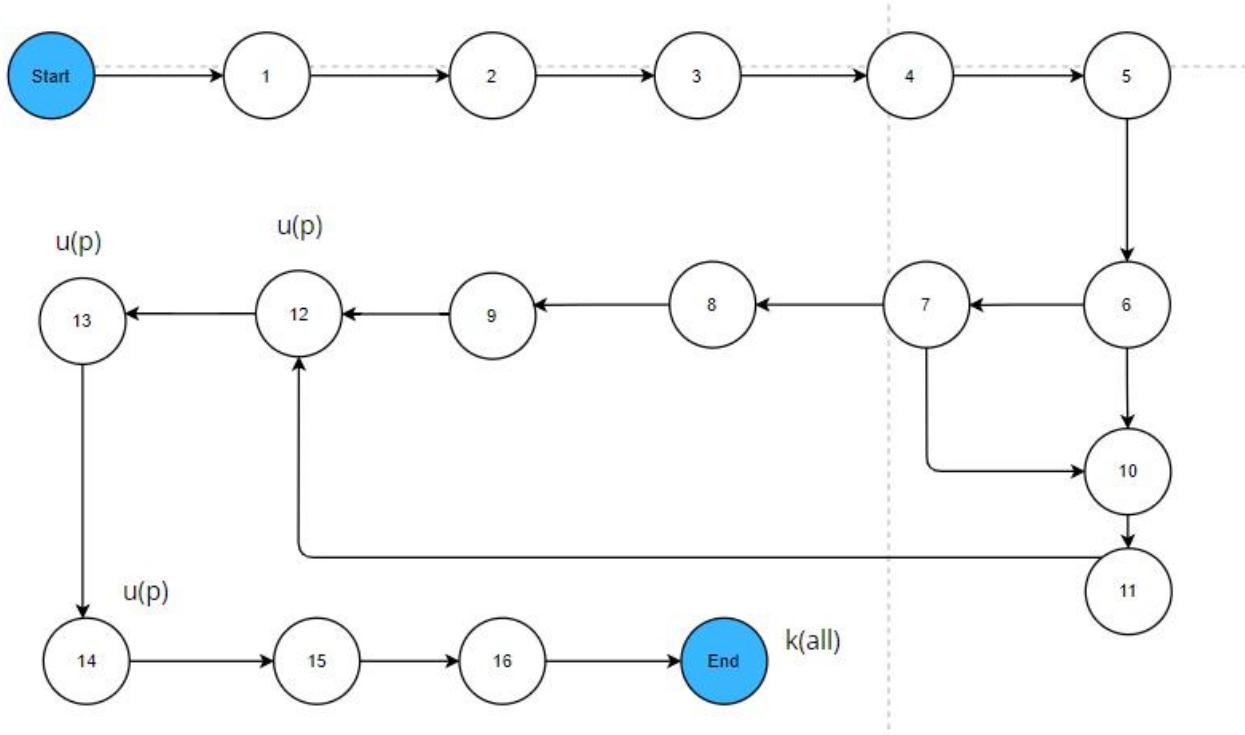
Kịch bản 1: ~uuuuuuk

Kịch bản 2: ~uuk

Kịch bản 3: ~uuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển $p(\text{avg})$:



Hình 6.14.p: Đồ thị đời sống biển $p(\text{avg})$ Đánh giá 1 sản phẩm

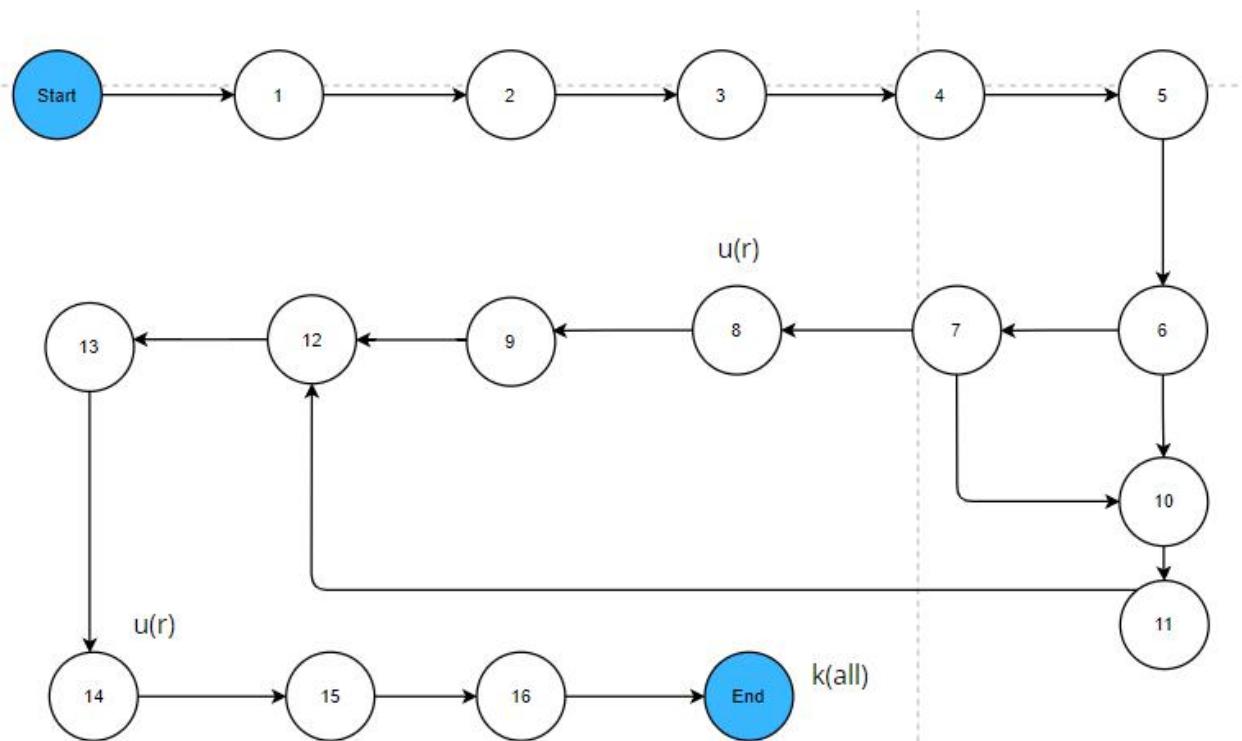
Kịch bản 1: ~uuuk

Kịch bản 2: ~uuuk

Kịch bản 3: ~uuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biến r (ratings):



Hình 6.14.r: Đồ thị đời sống biến r (ratings) Đánh giá 1 sản phẩm

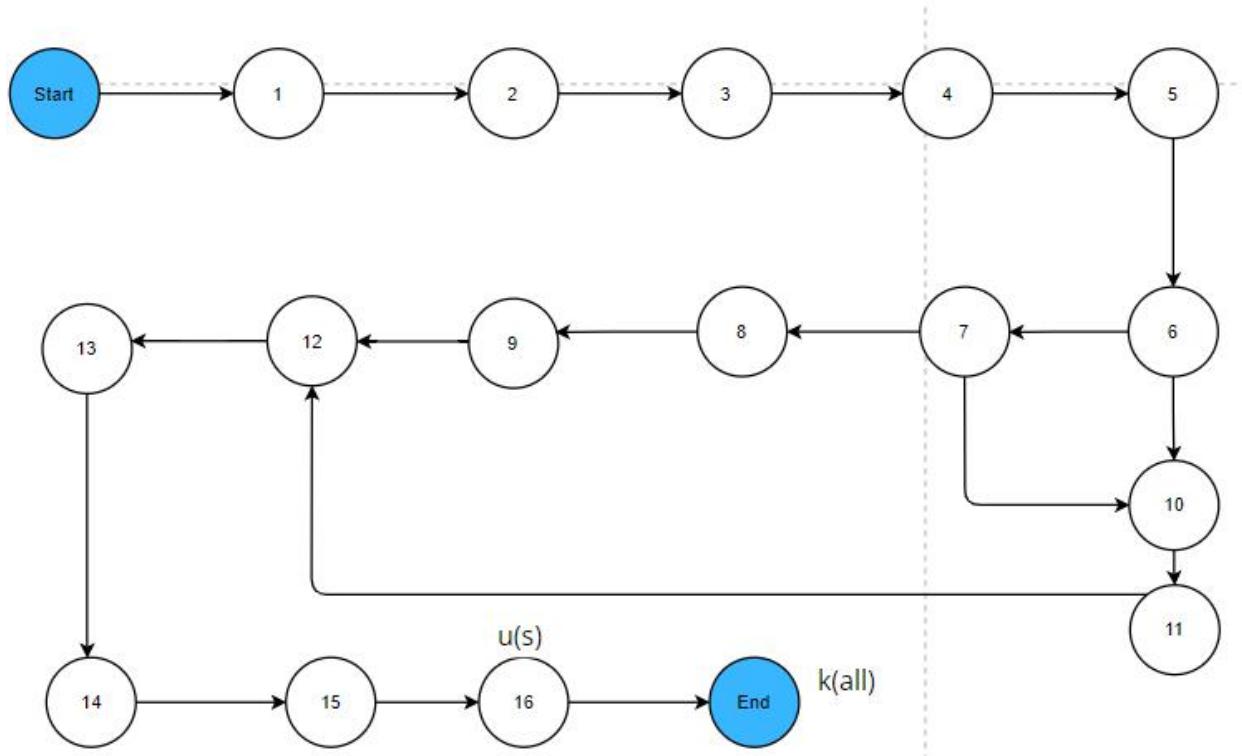
Kịch bản 1: ~uuuk

Kịch bản 2: ~uk

Kịch bản 3: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển s(success):



Hình 6.14.s: Đồ thị đời sống biển s(success) Đánh giá 1 sản phẩm

Kịch bản 1: ~uk

Kịch bản 2: ~uk

Kịch bản 3: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

16. Customer - Take an order

```
// Create A New Order
exports.newOrder = catchAsyncError(async (req, res, next/*(1)*)) => {
  const {
    shippingInfo,
    orderItems,
    paymentInfo,
    itemsPrice,
    taxPrice,
    shippingPrice,
    totalPrice,
  } = req.body; /*(2)*

  const order = await Order.create({
    shippingInfo,
    orderItems,
    paymentInfo,
    itemsPrice,
    taxPrice,
    shippingPrice,
    totalPrice,
  });
  res.status(201).json(order);
}
```

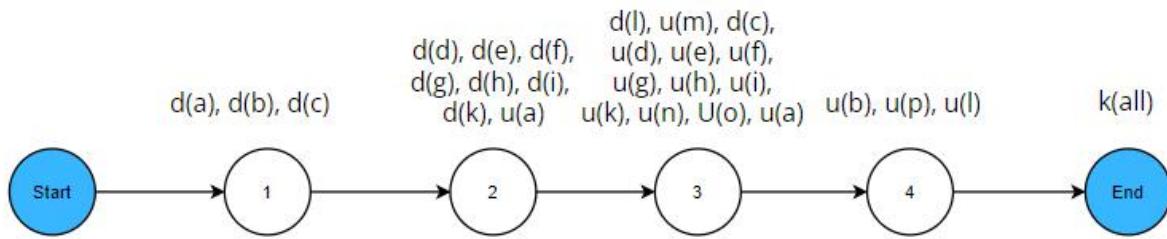
```

    shippingInfo,
    orderItems,
    paymentInfo,
    itemsPrice,
    taxPrice,
    shippingPrice,
    totalPrice,
    paidAt: Date.now(),
    user: req.user._id,
}); /*(3)*/

res.status(201).json({
  success: true,
  order,
}); /*(4)*/
);

```

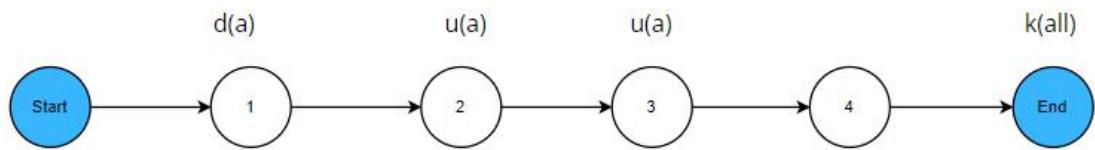
Table 6.16: Code Backend Đặt đơn hàng



Hình 6.16: Đồ thị dòng dữ liệu Đặt đơn hàng

Kiểm thử dòi sóng 15 biến: a(req), b(res), c(next), d(shippingInfo), e(orderItems), f(paymentInfo), g(itemsPrice), h(taxPrice), i(shippingPrice), k(totalPrice), l(order), m(Order), n(paidAt), o(user), p(success)

- Kiểm thử đời sóng biển a(req):

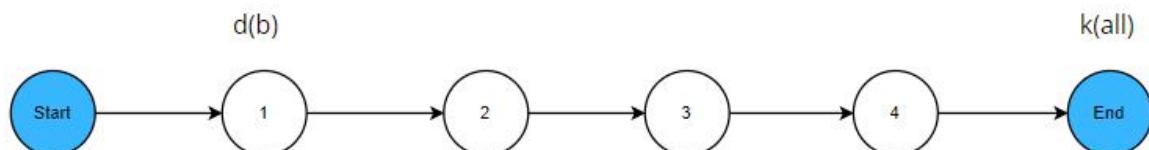


Hình 6.16.a: Đồ thị đời sóng biển a(req) Đặt đơn hàng

Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển b(res):

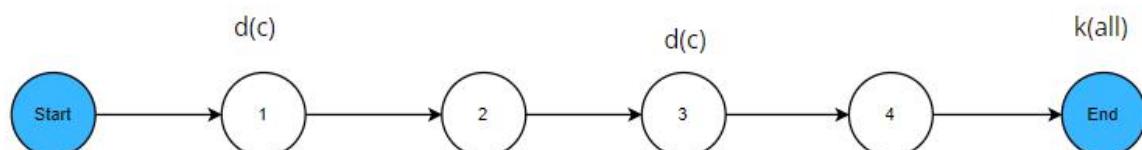


Hình 6.16.b: Đồ thị đời sóng biển b(res) Đặt đơn hàng

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biển c(next):

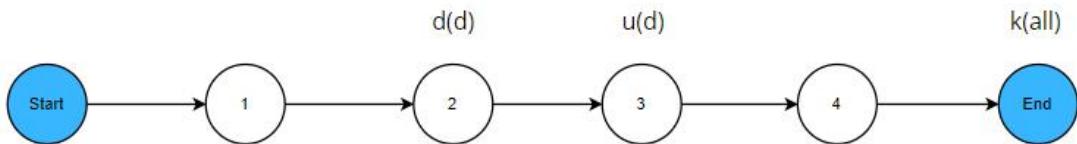


Hình 6.16.c: Đồ thị đời sóng biển c(next) Đặt đơn hàng

Kịch bản 1: ddk~

=> **Kết luận:** Cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sống biển d(shippingInfo):**

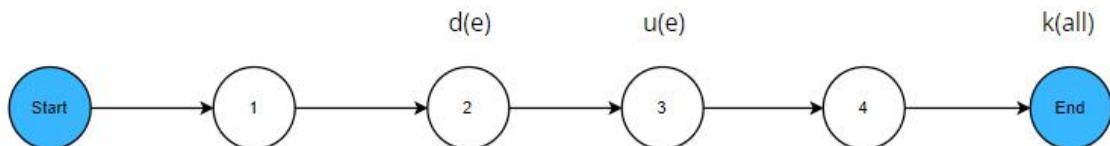


Hình 6.16.d: Đồ thị đời sống biển d(shippingInfo) Đặt đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sống biển e(orderItems):**

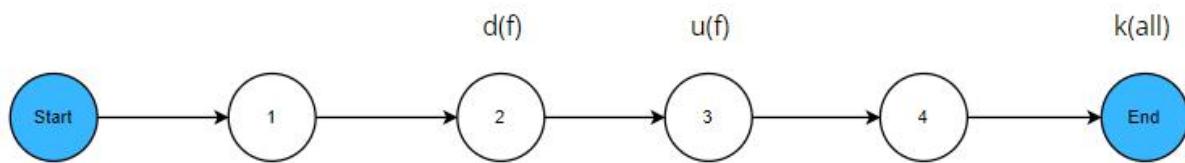


Hình 6.16.e: Đồ thị đời sống biển e(orderItems) Đặt đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sống biển f(paymentInfo):**

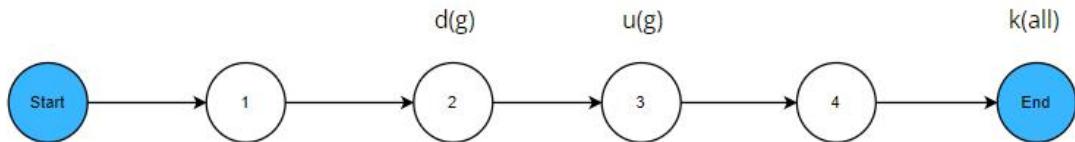


Hình 6.16.f: Đồ thị đời sống biển f(paymentInfo) Đặt đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòi hỏi biến g(itemsPrice):**

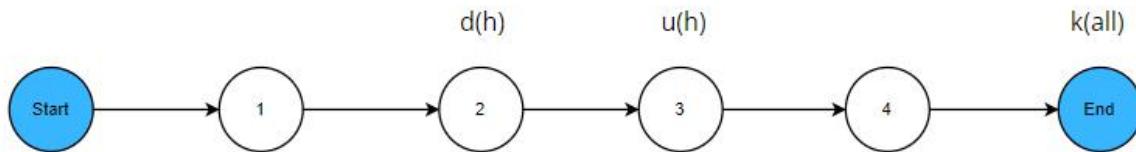


Hình 6.16.g: Đồ thị đòi hỏi biến g(itemsPrice) Đặt đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòi hỏi biến h(taxPrice):**

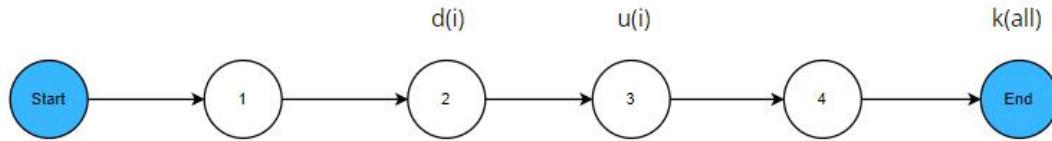


Hình 6.16.h: Đồ thị đòi hỏi biến h(taxPrice) Đặt đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòi hỏi biến i(shippingPrice):**

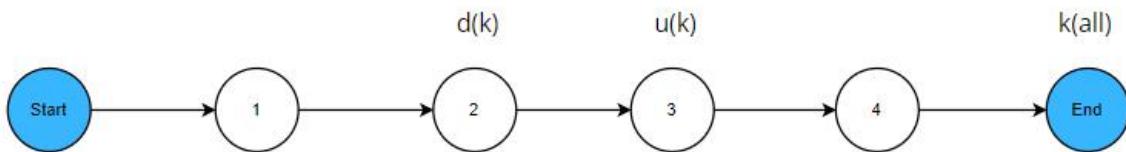


Hình 6.16.i: Đồ thị đòi hỏi biến i(shippingPrice) Đặt đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sống biến k(totalPrice):**

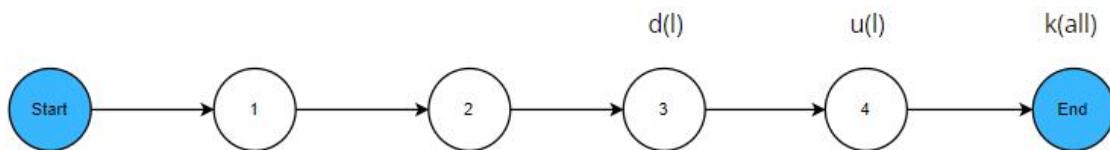


Hình 6.16.k: Đồ thị đời sống biến k (totalPrice) Đặt đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sống biến l(order):**

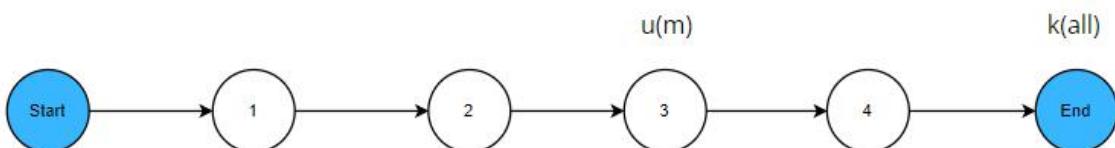


Hình 6.16.l: Đồ thị đời sống biến l (order) Đặt đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sống biến m(Order):**

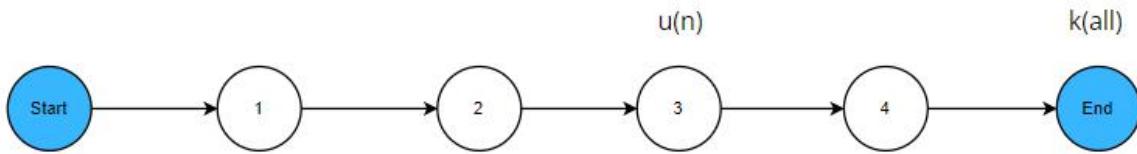


Hình 6.1.m: Đồ thị đời sống biến m (Order) Đặt đơn hàng

Kịch bản 1: uk~

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sống biến n (paidAt):

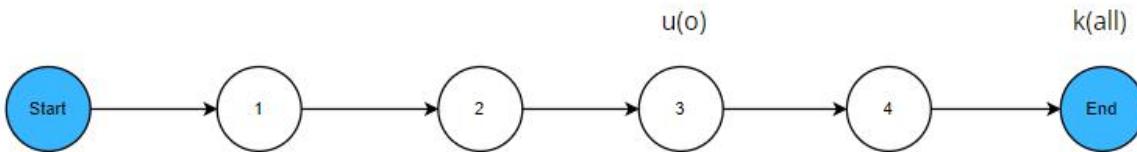


Hình 6.16.n: Đồ thị đời sống biến n (paidAt) Đặt đơn hàng

Kịch bản 1: uk~

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sống biến o (user):

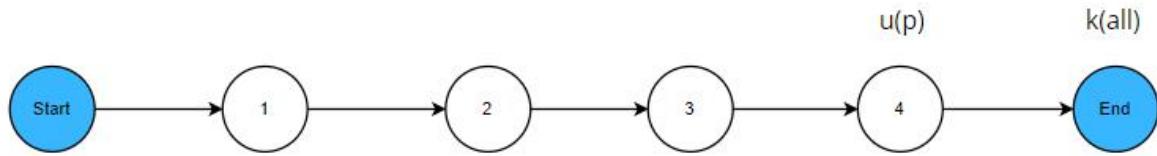


Hình 6.16.o: Đồ thị đời sống biến o (user) Đặt đơn hàng

Kịch bản 1: uk~

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sống biến p (success):



Hình 6.16.p: Đồ thị đời sống biến p (success) Đặt đơn hàng

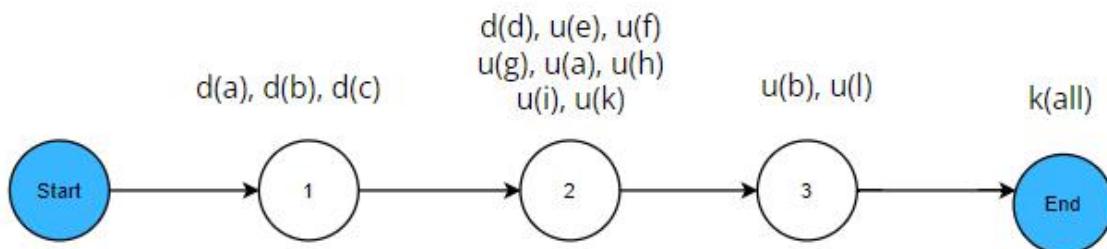
Kịch bản 1: uk~

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

17. Customer - Check out

```
exports.processPayment = catchAsyncError(async (req, res, next) => {
  const myPayment = await stripe.paymentIntents.create({
    amount: req.body.amount,
    currency: "vnd",
    metadata: {
      company: "Gatitos Group",
    },
  });
  /*(2)*/
  res
    .status(200)
    .json({ success: true, client_secret: myPayment.client_secret });
}); /*(3)*/
```

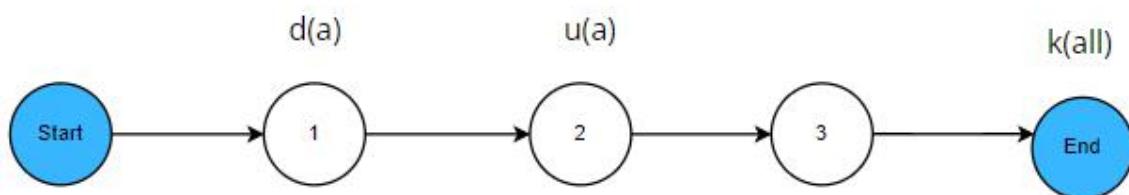
Table 6.17: Code Backend Thanh toán đơn hàng



Hình 6.17: Đồ thị dòng dữ liệu Thanh toán đơn hàng

Kiểm thử đời sống 11 biến: a(req), b(res), c(next), d(myPayment), e(stripe), f(paymentIntents), g(amount), h(currency), i(metadata), k(company), l(success)

- **Kiểm thử đời sống biến a(req):**

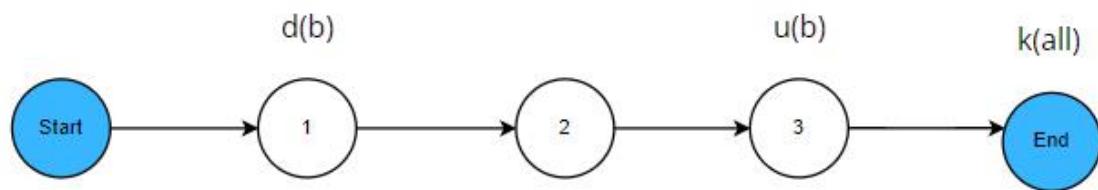


Hình 6.17.a: Đồ thị đợi sóng biến $a(req)$ Thanh toán đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đợi sóng biến $b(res)$:**

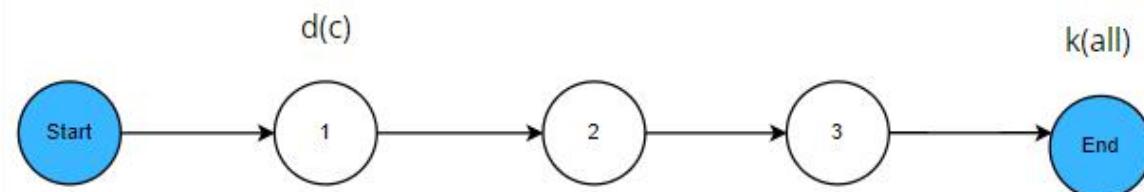


Hình 6.17.b: Đồ thị đợi sóng biến $b(res)$ Thanh toán đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đợi sóng biến $c(next)$:**

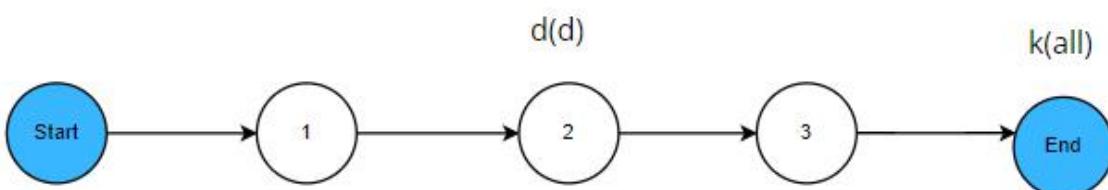


Hình 6.17.c: Đồ thị đợi sóng biến $c(next)$ Thanh toán đơn hàng

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đợi sóng biến $d(myPayment)$:**

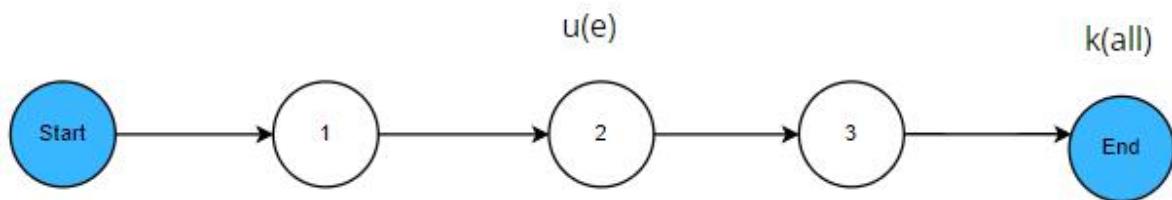


Hình 6.17.d: Đồ thị đòn bẩy biến $d(myPayment)$ Thanh toán đơn hàng

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòn bẩy biến $e(stripe)$:**

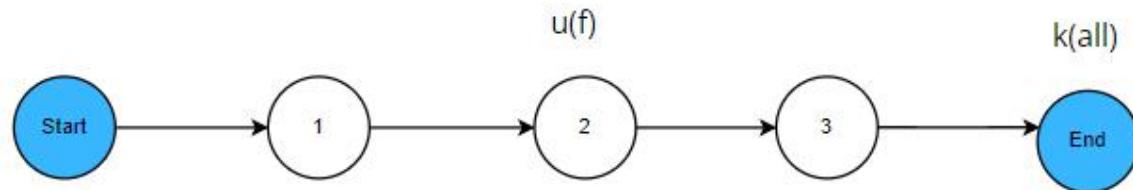


Hình 6.17.e: Đồ thị đòn bẩy biến $e(stripe)$ Thanh toán đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòn bẩy biến $f(paymentIntents)$:**

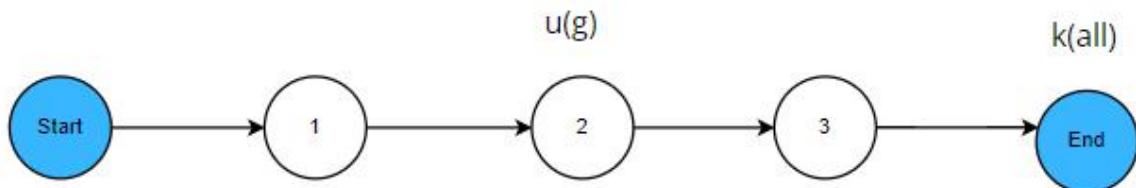


Hình 6.17.f: Đồ thị đòn bẩy biến $f(paymentIntents)$ Thanh toán đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòn bẩy biến $g(amount)$:**

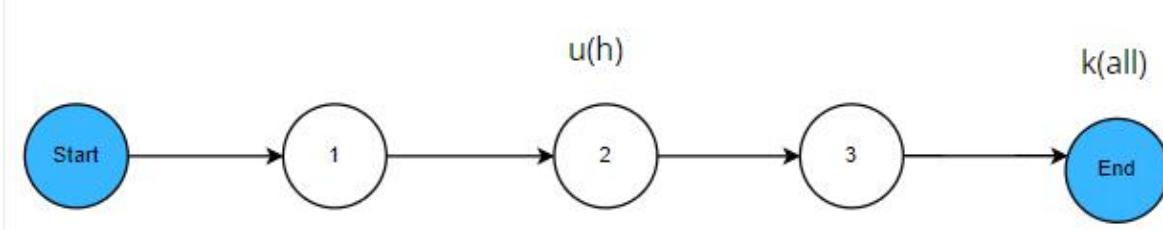


Hình 6.17.g: Đồ thị đòi hỏi biến g(amount) Thanh toán đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòi hỏi biến h(currency):**

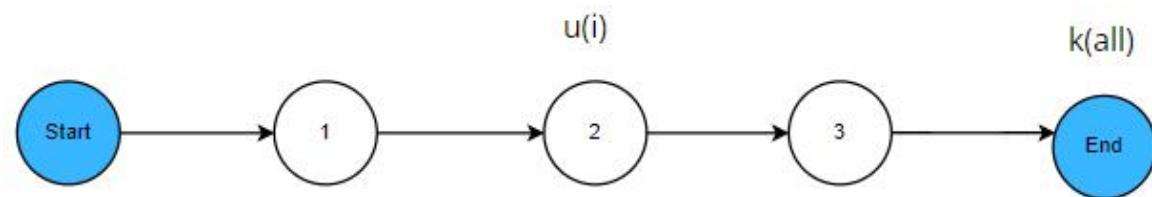


Hình 6.17.h: Đồ thị đòi hỏi biến h(currency) Thanh toán đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòi hỏi biến i(metadata):**

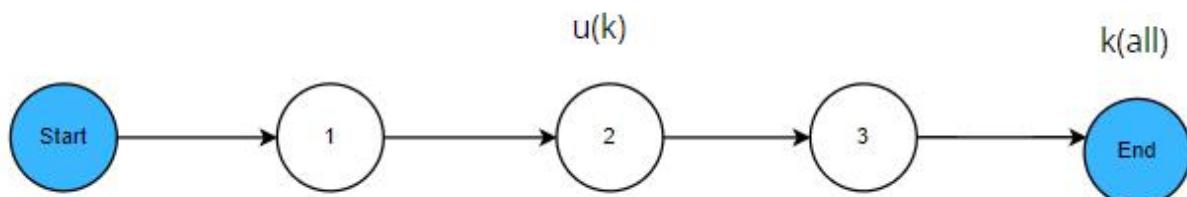


Hình 6.17.i: Đồ thị đòi hỏi biến i(metadata) Thanh toán đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đòi hỏi biến k(company):**

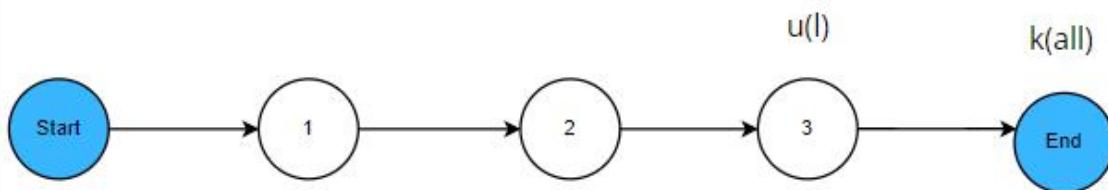


Hình 6.17.k: Đồ thị đời sống biến k(company) Thanh toán đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sống biến l(success):**



Hình 6.17.l: Đồ thị đời sống biến l(success) Thanh toán đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

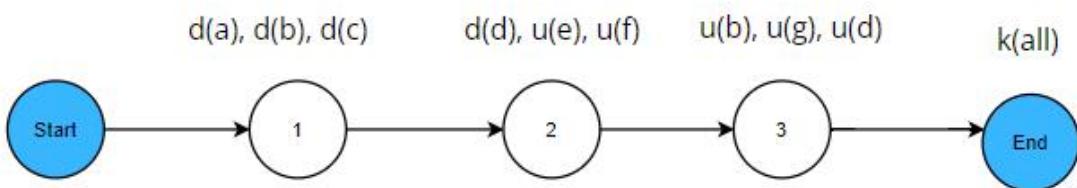
18. Customer - See all orders

```

//Get logged in user Orders
exports.myOrders = catchAsyncError(async (req, res, next/*(1)*)) => {
  const orders = await Order.find({ user: req.user._id }); /*(2)*

  res.status(200).json({
    success: true,
    orders,
  }); /*(3)*
});
  
```

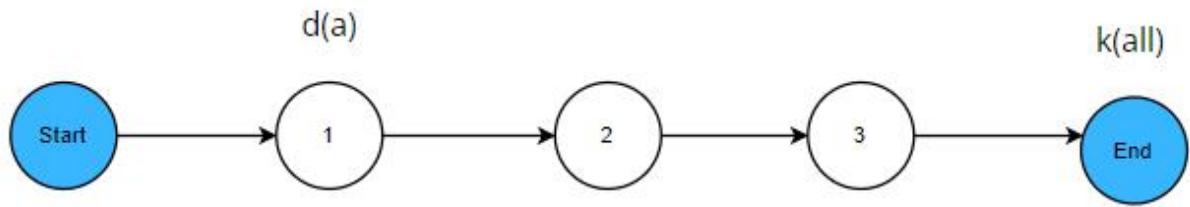
Table 6.18: Code Backend Xem tất cả các đơn hàng



Hình 6.18: Đồ thị dòng dữ liệu Xem tất cả các đơn hàng

Kiểm thử đời sống 7 biến: a(req), b(res), c(next), d(orders), e(Order), f(user), g(success)

- Kiểm thử đời sóng biến $a(req)$:

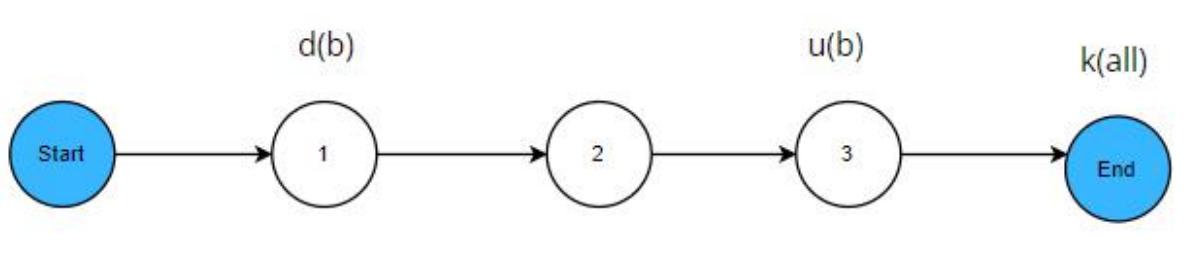


Hình 6.18.a: Đồ thị đời sóng biến $a(req)$ Xem tất cả các đơn hàng

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biến $b(res)$:

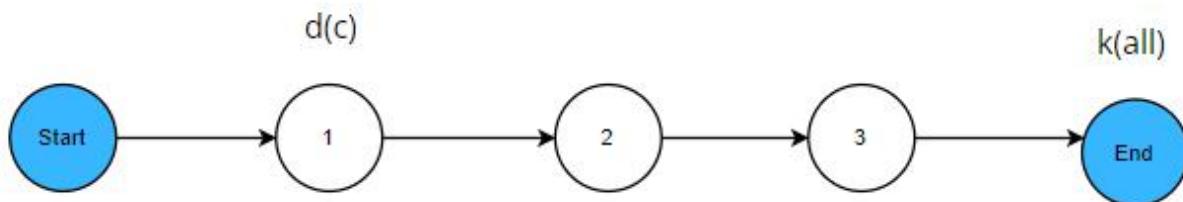


Hình 6.18.b: Đồ thị đời sóng biến $b(res)$ Xem tất cả các đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sóng biến $c(next)$:

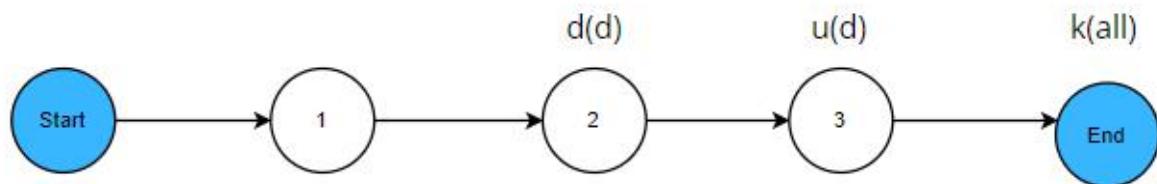


Hình 6.18.c: Đồ thị đời sóng biến $c(next)$ Xem tất cả các đơn hàng

Kịch bản 1: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sóng biến d(orders):**

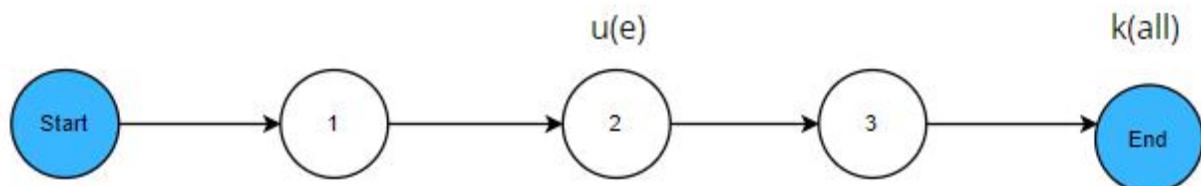


Hình 6.18.d: Đồ thị đời sóng biến $d(orders)$ Xem tất cả các đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- **Kiểm thử đời sóng biến e(Order):**

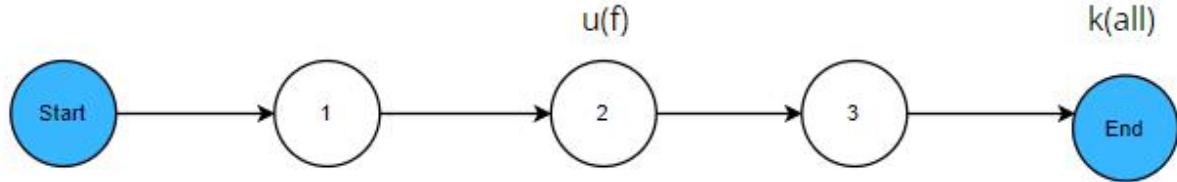


Hình 6.18.e: Đồ thị đời sóng biến $e(Order)$ Xem tất cả các đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sống biến $f(user)$:

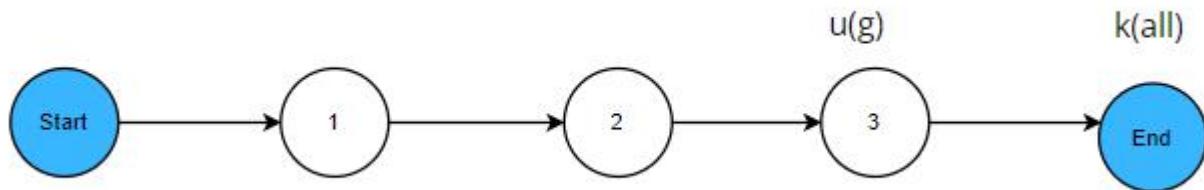


Hình 6.18.f: Đồ thị đời sống biến $f(user)$ Xem tất cả các đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

- Kiểm thử đời sống biến $g(success)$:



Hình 6.18.g: Đồ thị đời sống biến $g(success)$ Xem tất cả các đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường

19. Customer - See a single order

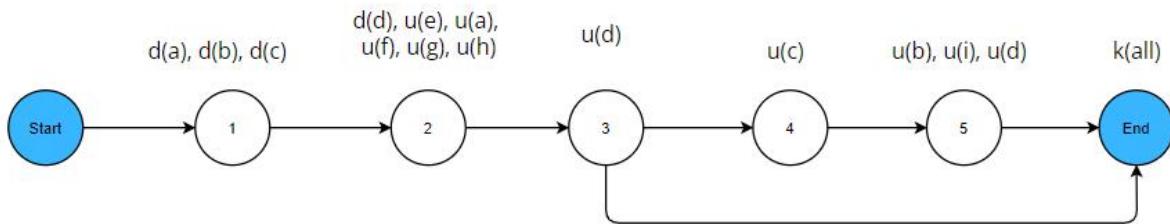
```
//Get A Single Order
exports.getSingleOrder = catchAsyncError(async (req, res, next) => {
  const order = await Order.findById(req.params.id).populate(
    "user",
    "name email"
  ); /*(2)*/
  if (!order /*(3)*{
    return next(new ErrorHandler("Order not found with this Id", 404)); /*(4)*
  }
```

```

res.status(200).json({
  success: true,
  order,
}); /*(5)*/
});

```

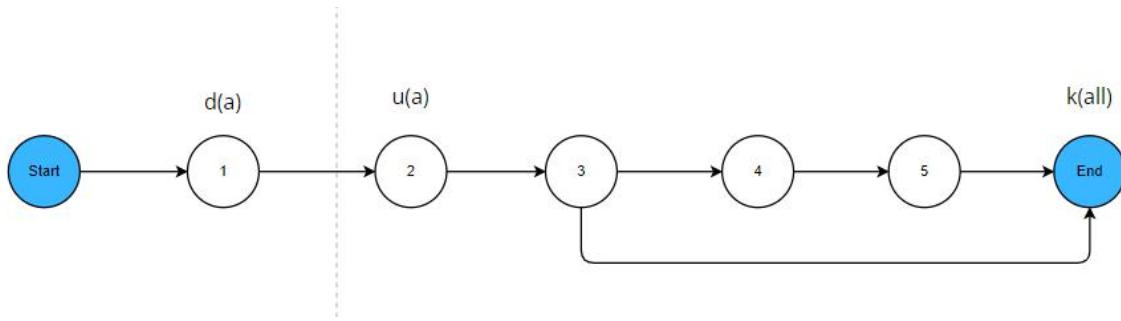
Table 6.19: Code Backend Xem thông tin 1 đơn hàng



Hình 6.19: Đồ thị dòng dữ liệu Xem thông tin 1 đơn hàng

Kiểm thử dòng sống 9 biến: a(req), b(res), c(next), d(order), e(Order), f(id), g(user), h(name email), i(success)

- Kiểm thử dòng sống biến a(req):



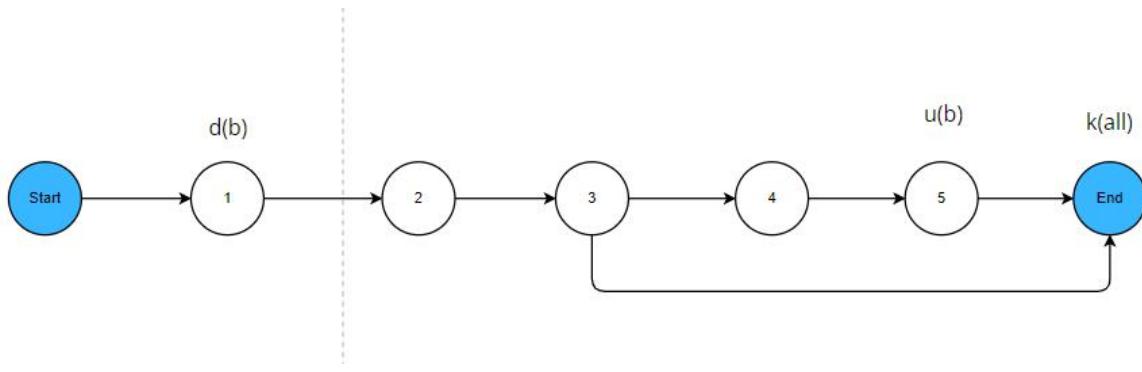
Hình 6.19.a: Đồ thị dòng sống biến a(req) Xem thông tin 1 đơn hàng

Kịch bản 1: ~duk

Kịch bản 2: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sóng biến $b(res)$:



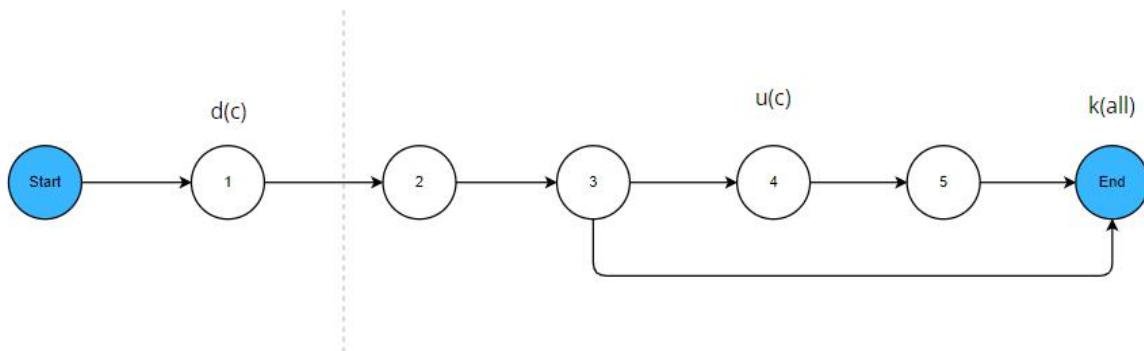
Hình 6.19.b: Đồ thị đời sóng biến $b(res)$ Xem thông tin 1 đơn hàng

Kịch bản 1: ~duk

Kịch bản 2: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sóng biến $c(next)$:



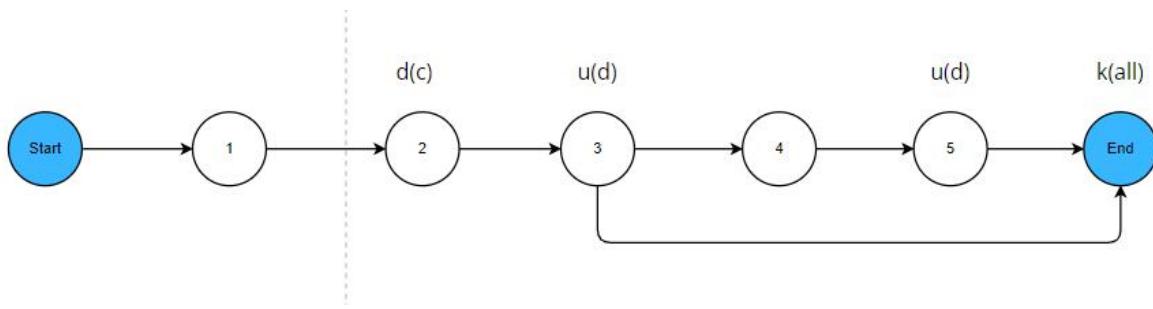
Hình 6.19.c: Đồ thị đời sóng biến $c(next)$ Xem thông tin 1 đơn hàng

Kịch bản 1: ~duk

Kịch bản 2: ~dk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biến $d(order)$:



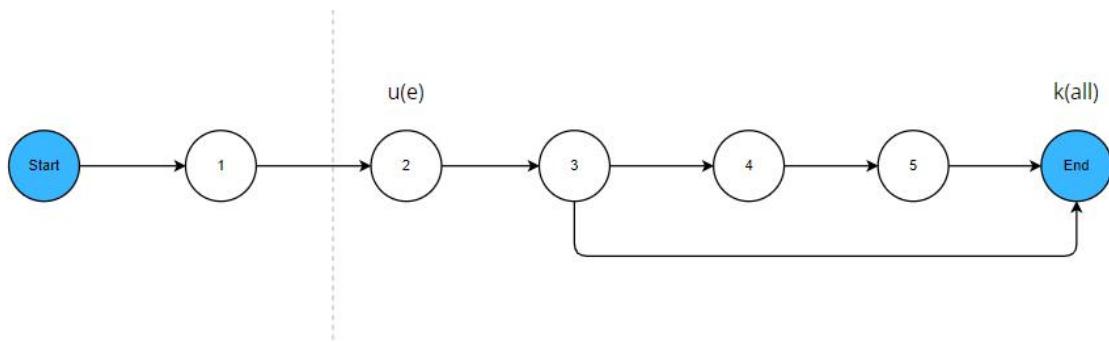
Hình 6.19.d: Đồ thị đời sống biến $d(order)$ Xem thông tin 1 đơn hàng

Kịch bản 1: ~duuk

Kịch bản 2: ~duk

=> Kết luận: Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biến $e(Order)$:



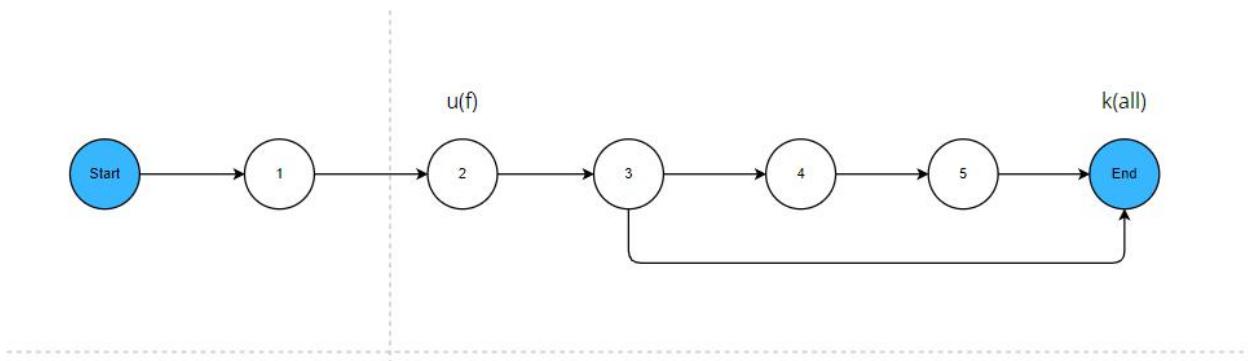
Hình 6.19.e: Đồ thị đời sống biến $e(Order)$ Xem thông tin 1 đơn hàng

Kịch bản 1: ~uk

Kịch bản 2: ~uk

=> Kết luận: Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển $f(id)$:



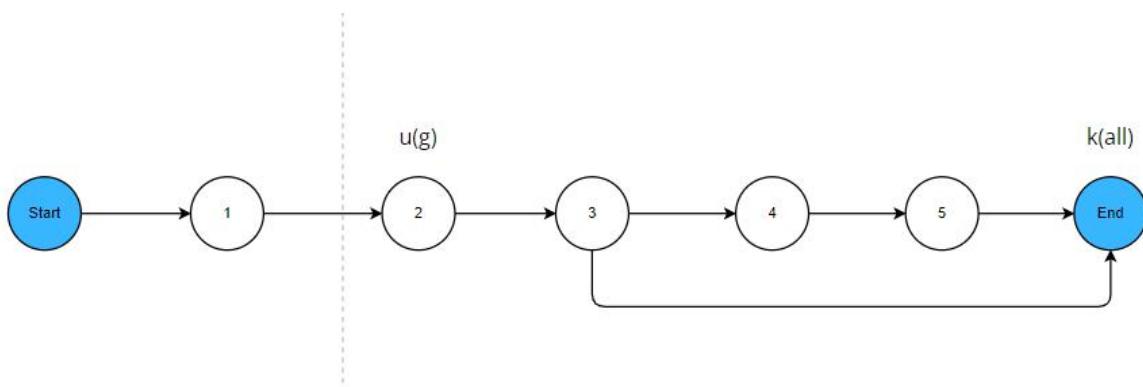
Hình 6.19.f: Đồ thị đời sống biển $f(id)$ Xem thông tin 1 đơn hàng

Kịch bản 1: ~uk

Kịch bản 2: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển $g(user)$:



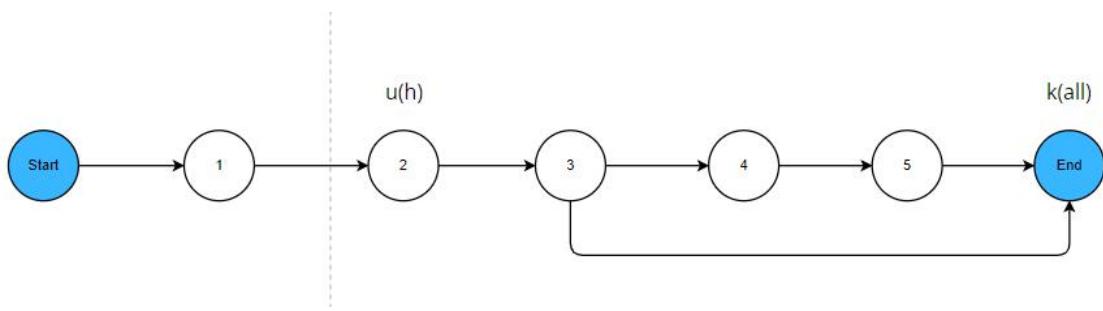
Hình 6.19.g: Đồ thị đời sống biển $g(user)$ Xem thông tin 1 đơn hàng

Kịch bản 1: ~uk

Kịch bản 2: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển h(name email):



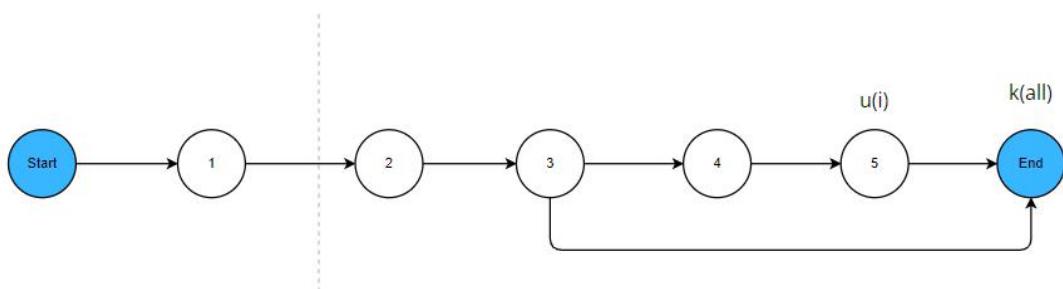
Hình 6.19.h: Đồ thị đời sống biển h(name email) Xem thông tin 1 đơn hàng

Kịch bản 1: ~uk

Kịch bản 2: ~uk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản hoạt động bất thường

- Kiểm thử đời sống biển i(success):



Hình 6.19.i: Đồ thị đời sống biển ai(success) Xem thông tin 1 đơn hàng

Kịch bản 1: ~uk

Kịch bản 2: ~k

=> **Kết luận:** Cặp đôi nào của Kịch bản 2 hoạt động bất thường

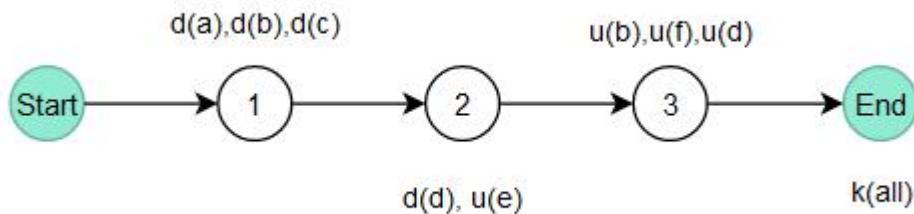
20. Admin - See All Users

```

// Get all users (admin)
exports.getAllUser = catchAsyncError(async (req, res, next) => {
  const users = await User.find(); /*(2)*/
  res.status(200).json({
    success: true,
    users,
  }); /*(3)*/ 
}
  
```

});

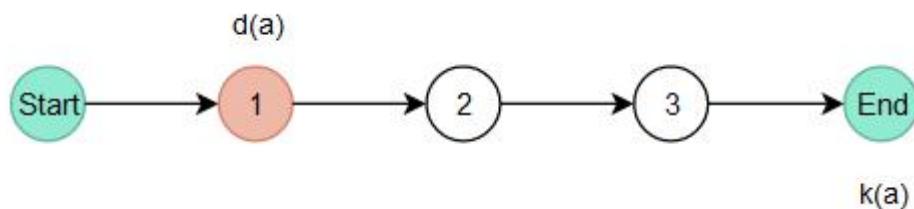
Table 6.20: Code Backend Xem tất cả các người dùng



Hình 6.20: Đồ thị dòng dữ liệu Xem tất cả các người dùng

Kiểm thử dòng sóng 6 biến: a(req), b(res), c(next), d(users), e(User), f(success)

- Kiểm thử dòng sóng biến a(req):

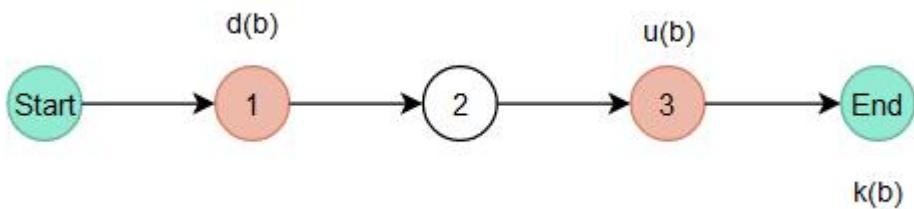


Hình 6.20.a: Đồ thị dòng sóng biến a(req) Xem tất cả các người dùng

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử dòng sóng biến b(res):

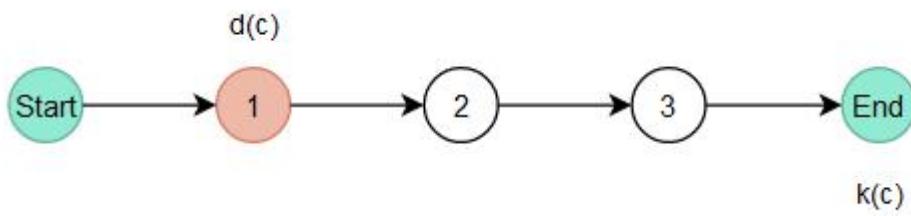


Hình 6.20.b: Đồ thị dòng sóng biến b(res) Xem tất cả các người dùng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biển c(next):**

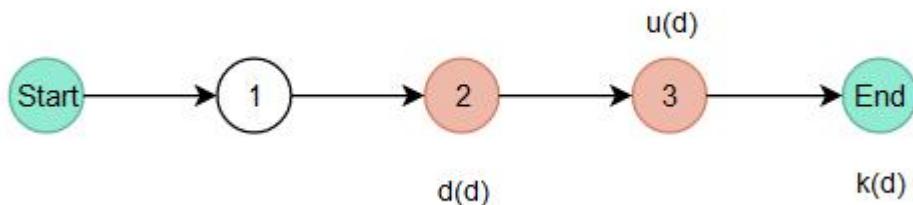


Hình 6.20.c: Đồ thị đời sóng biển $c(next)$ Xem tất cả các người dùng

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- **Kiểm thử đời sóng biển d(users):**

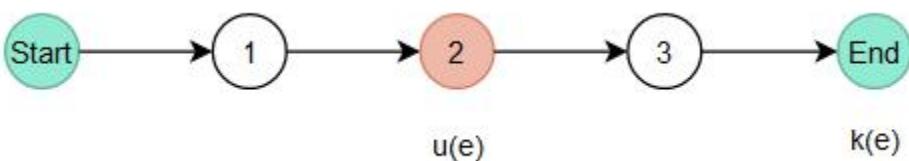


Hình 6.20.d: Đồ thị đời sóng biển $d(users)$ Xem tất cả các người dùng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sóng biển e(User):**

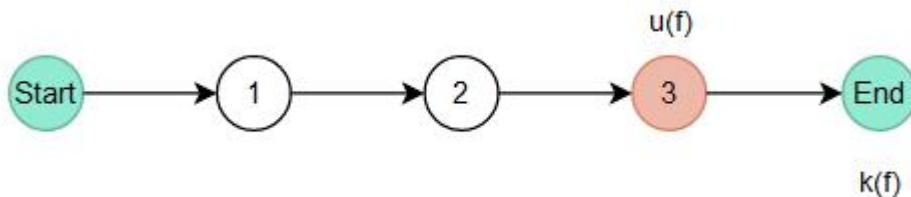


Hình 6.20.e: Đồ thị đời sóng biển $e(User)$ Xem tất cả các người dùng

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- **Kiểm thử đời sóng biển f(success):**



Hình 6.20.f: Đồ thị đời sóng biển f(success) Xem tất cả các người dùng

Kịch bản 1: $\sim u k$

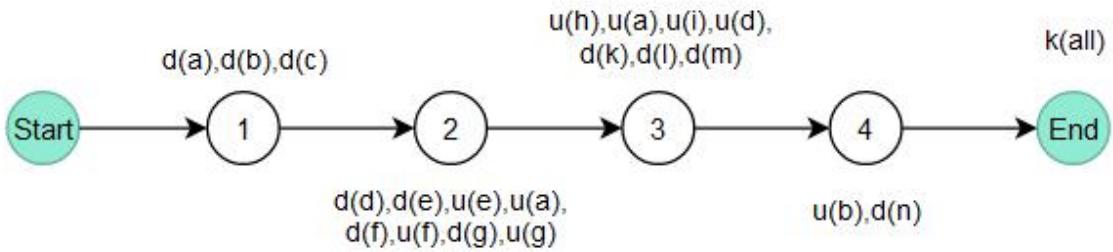
=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

21. Admin - Update User Role

```

// Update User Role (admin)
exports.updateUserRole = catchAsyncError(async (req, res, next/*(1)*/) => {
    const newUserData = {
        name: req.body.name,
        email: req.body.email,
        role: req.body.role,
    }; /*(2)*/
    await User.findByIdAndUpdate(req.params.id, newUserData, {
        new: true,
        runValidators: true,
        useFindAndModify: false,
   }); /*(3)*/
    res.status(200).json({
        success: true,
   }); /*(4)*/
});
  
```

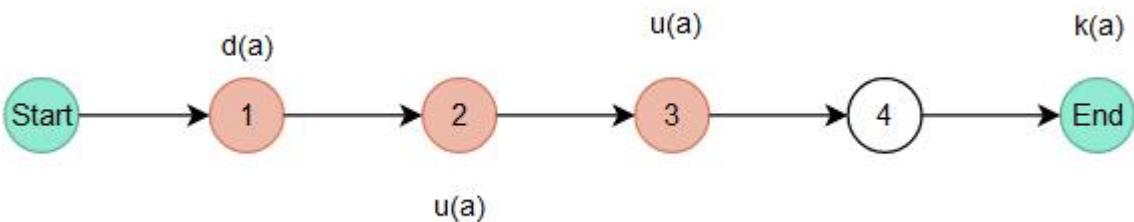
Table 6.21: Code Backend Nâng cấp quyền hạn người dùng



Hình 6.21: Đồ thị dòng dữ liệu Nâng cấp quyền hạn người dùng

Kiểm thử đời sống 13 biến: $a(\text{req})$, $b(\text{res})$, $c(\text{next})$, $d(\text{newUserData})$, $e(\text{name})$, $f(\text{email})$, $g(\text{role})$, $h(\text{User})$, $i(\text{id})$, $k(\text{new})$, $l(\text{runValidators})$, $m(\text{useFindAndModify})$, $n(\text{success})$

- **Kiểm thử đời sống biến $a(\text{req})$:**

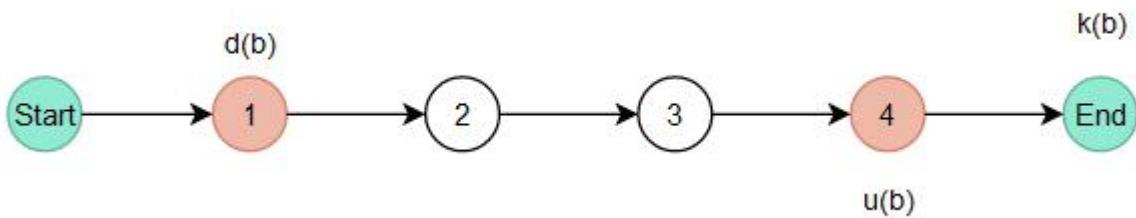


Hình 6.21.a: Đồ thị đời sống biến $a(\text{req})$ Đăng ký tài khoản

Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biến $b(\text{res})$:**

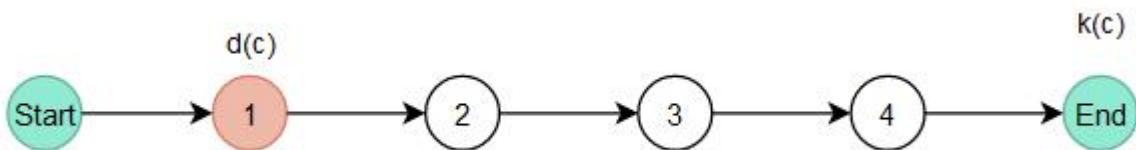


Hình 6.21.b: Đồ thị đời sống biến $b(\text{res})$ Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến $c(next)$:

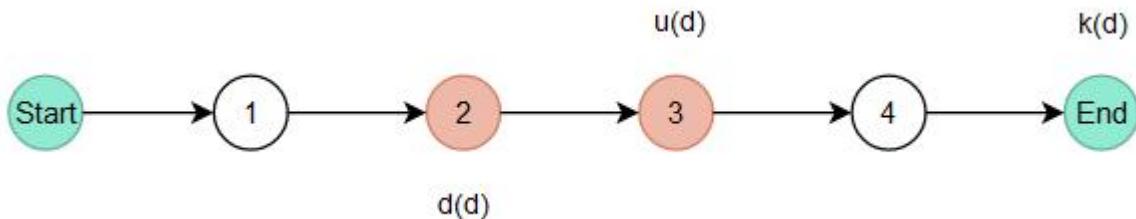


Hình 6.21.c: Đồ thị đời sống biến $c(next)$ Đăng ký tài khoản

Kịch bản 1: $\sim dk$

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến $d(userData)$:

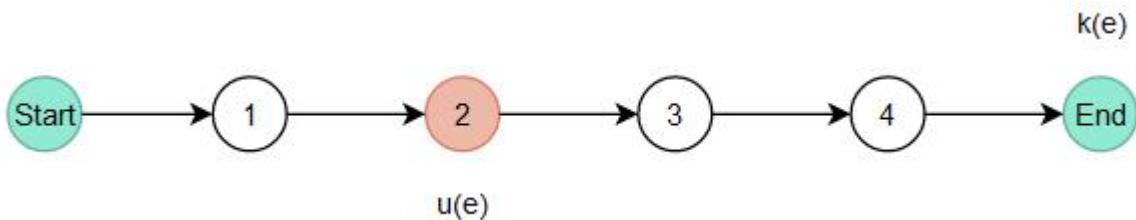


Hình 6.21.d: Đồ thị đời sống biến $d(userData)$ Đăng ký tài khoản

Kịch bản 1: $\sim du$

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến $e(name)$:

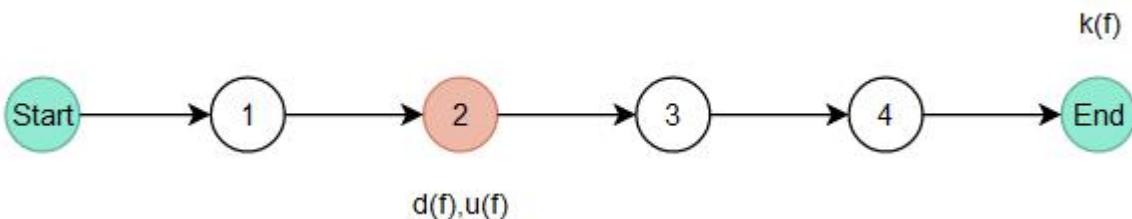


Hình 6.21.e: Đồ thị đời sống biến $e(name)$ Đăng ký tài khoản

Kịch bản 1: $\sim uk$

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biển $f(\text{email})$:

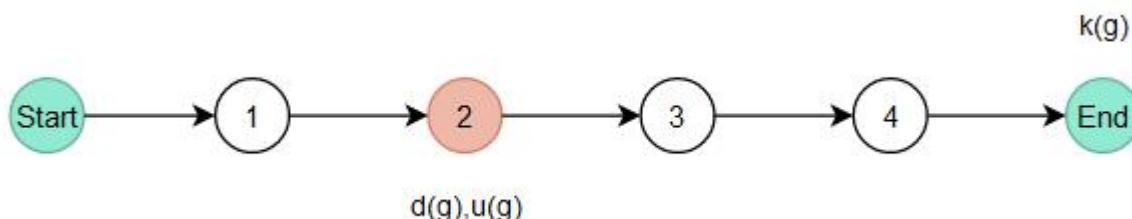


Hình 6.21.f: Đồ thị đời sóng biển $f(\text{email})$ Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển $g(\text{role})$:

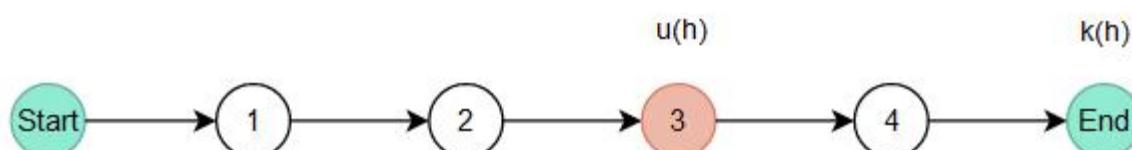


Hình 6.21.g: Đồ thị đời sóng biển $g(\text{role})$ Đăng ký tài khoản

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển $h(\text{User})$:

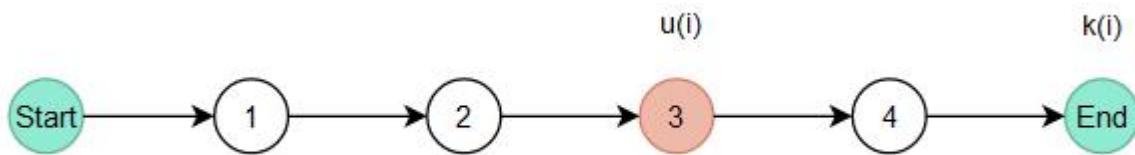


Hình 6.21.h: Đồ thị đời sóng biển $h(\text{User})$ Đăng ký tài khoản

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biến $i(id)$:

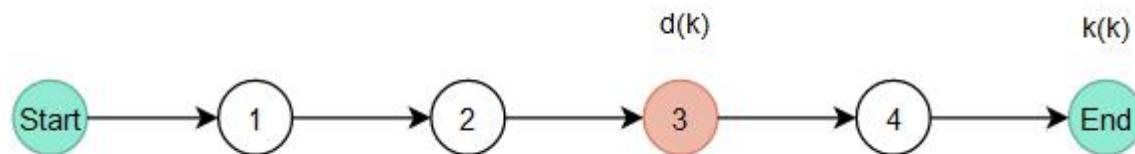


Hình 6.21.i: Đồ thị đời sống biến $i(id)$ Đăng ký tài khoản

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biến $k(new)$:

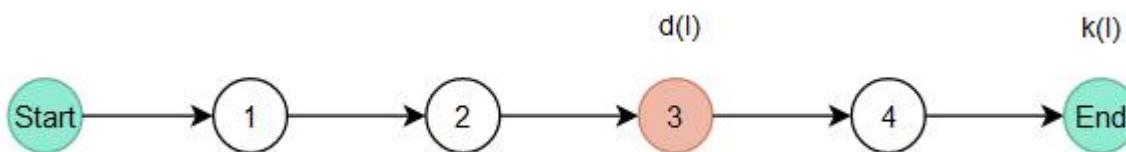


Hình 6.21.k: Đồ thị đời sống biến $k(new)$ Đăng ký tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến $l(runValidators)$:

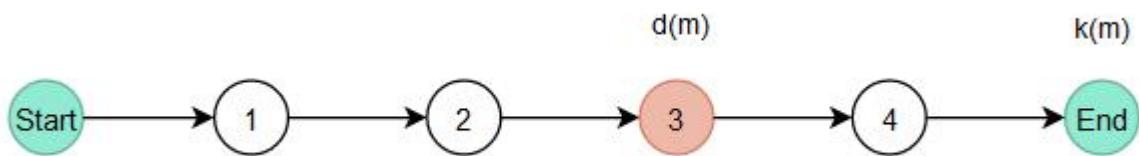


Hình 6.21.l: Đồ thị đời sống biến $l(runValidators)$ Đăng ký tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến m(useFindAndModify):

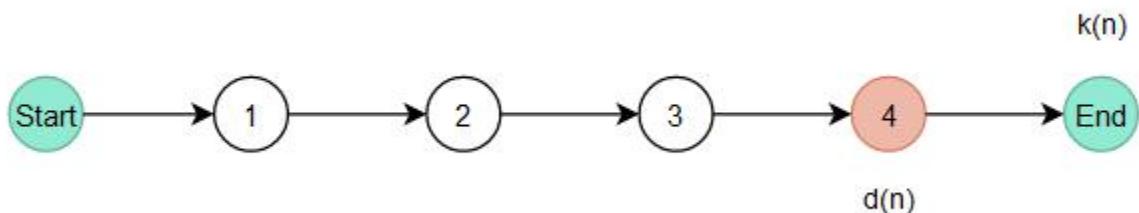


Hình 6.21.m: Đồ thị đời sống biến m(useFindAndModify) Đăng ký tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến n(success):



Hình 6.21.n: Đồ thị đời sống biến n(success) Đăng ký tài khoản

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

22. Admin - Delete Users

```
// Delete User (admin)
exports.deleteUser = catchAsyncError(async (req, res, next/*(1)*/) => {
  const user = await User.findById(req.params.id); /*(2)*/
  if (!user /*(3)*/) {
    return next(
      new ErrorHandler(`User does not exist with Id: ${req.params.id}`, 400)
    ); /*(4)*/
  }
  const imgId = user.avatar.public_id; /*(5)*/
```

```

await cloudinary.v2.uploader.destroy(imageId); /*(6)*/

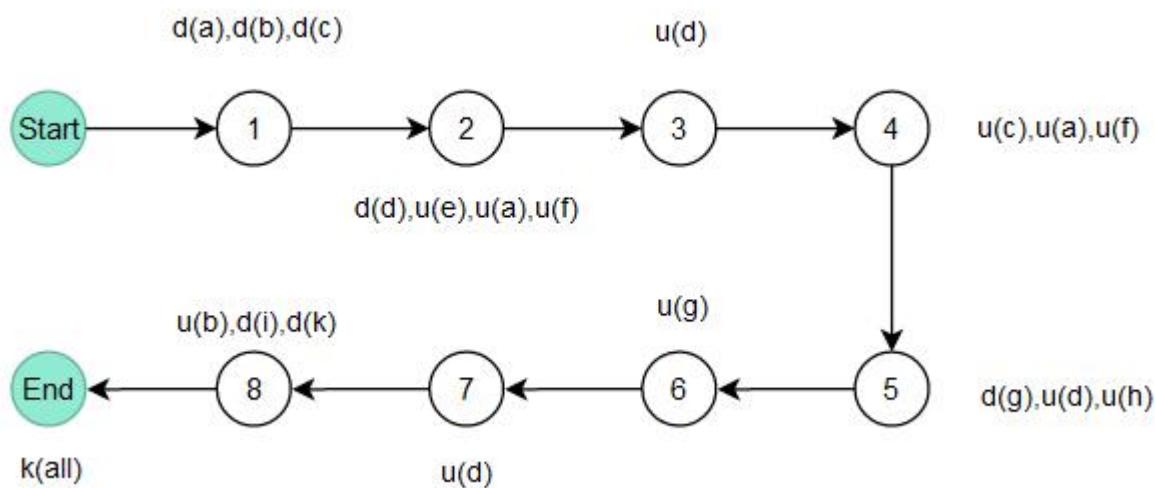
await user.remove(); /*(7)*/

res.status(200).json({
  success: true,
  message: "User deleted successfully",
}); /*(8)*/

});

```

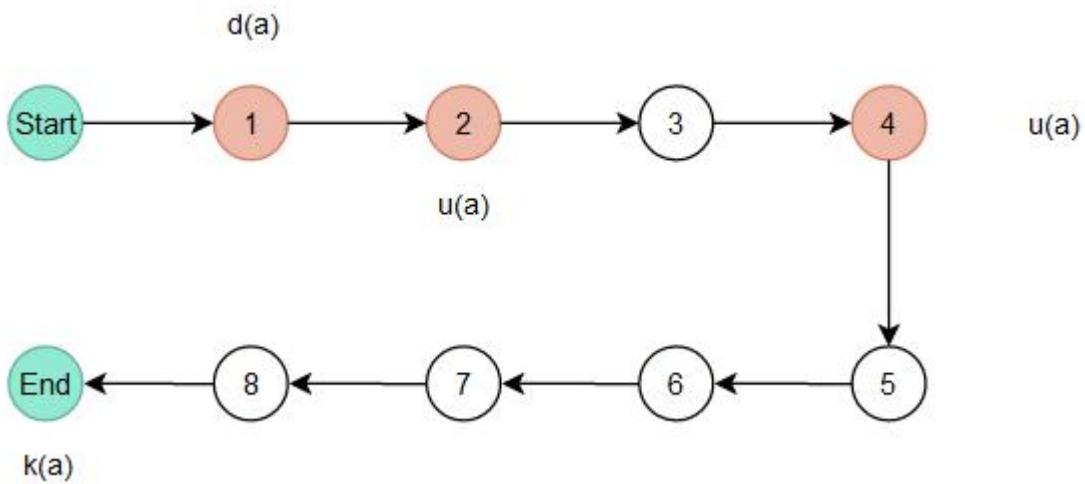
Table 6.22: Code Backend Xóa người dùng



Hình 6.22: Đồ thị dòng dữ liệu Nâng cấp quyền hạn người dùng

Kiểm thử đời sống 10 biến: a(req), b(res), c(next), d(user), e(User), f(id), g(imageId), h/avatar), i(success), k(message)

- Kiểm thử đời sóng biển a(req):

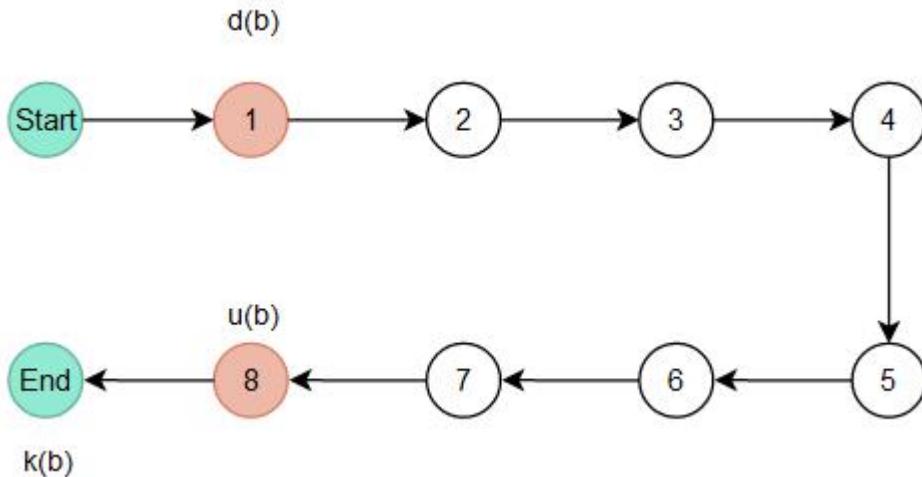


Hình 6.22.a: Đồ thị đời sóng biển a(req) Nâng cấp quyền hạn người dùng

Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển b(res):

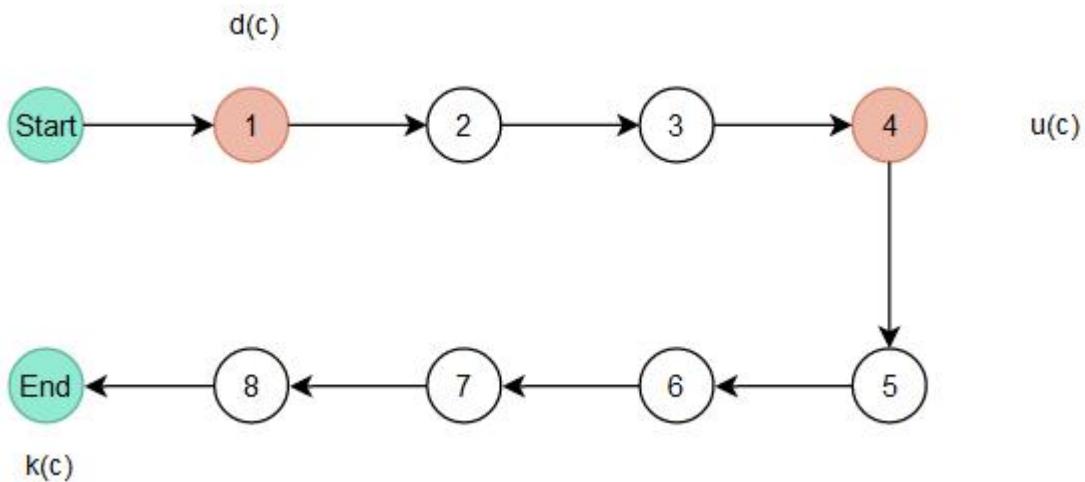


Hình 6.22.b: Đồ thị đời sóng biển b(res) Nâng cấp quyền hạn người dùng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến $c(next)$:

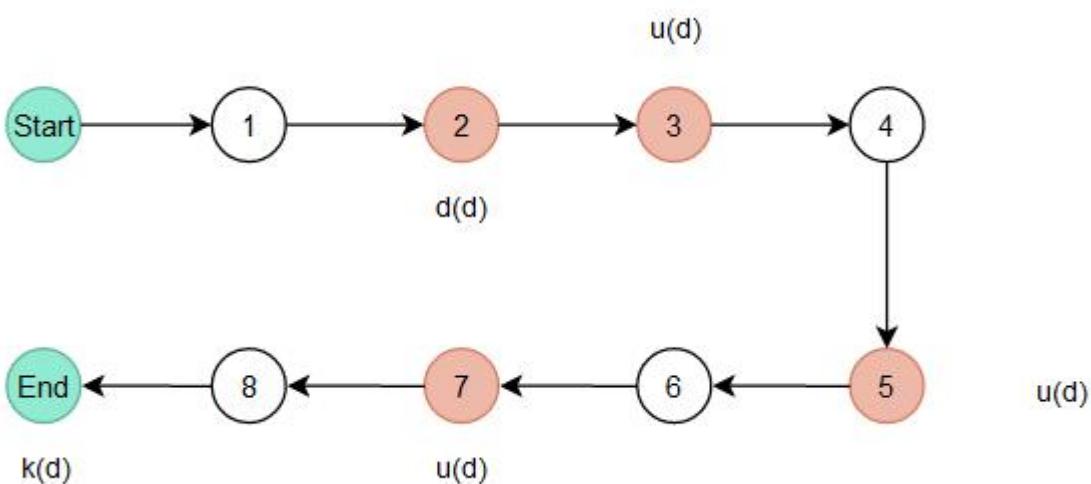


Hình 6.22.c: Đồ thị đời sống biến $c(next)$ Nâng cấp quyền hạn người dùng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến $d(user)$:

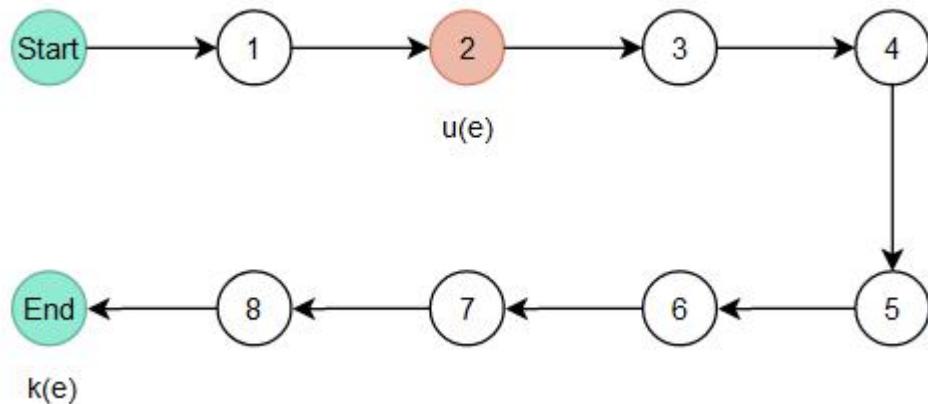


Hình 6.22.d: Đồ thị đời sống biến $d(user)$ Nâng cấp quyền hạn người dùng

Kịch bản 1: ~duuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển $e(User)$:

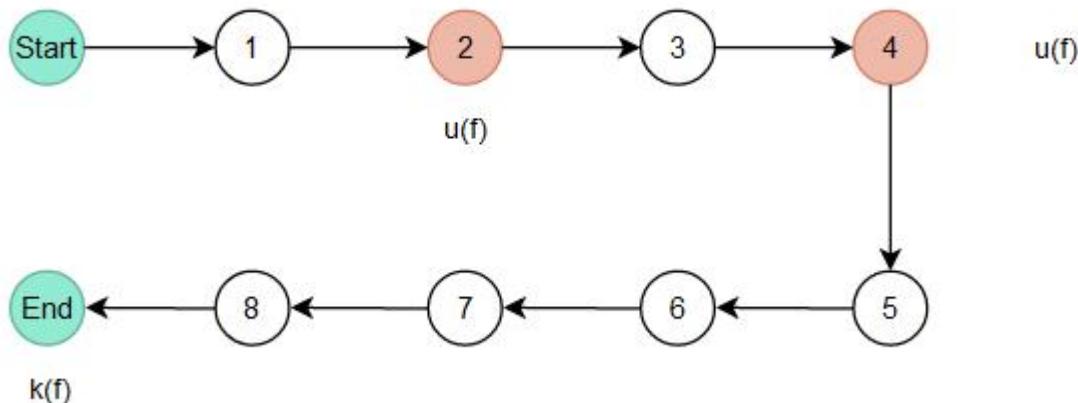


Hình 6.22.e: Đồ thị đời sóng biển $e(User)$ Nâng cấp quyền hạn người dùng

Kịch bản 1: $\sim u k$

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biển $f(id)$:

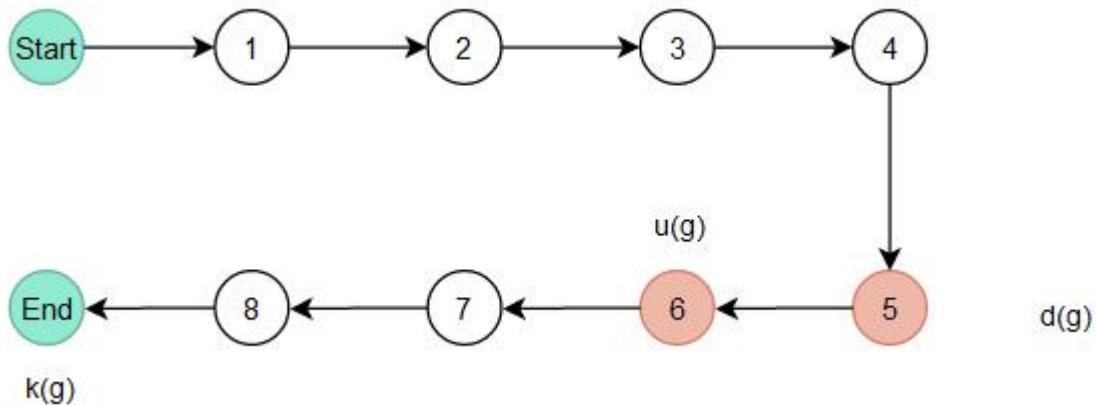


Hình 6.22.f: Đồ thị đời sóng biển $f(id)$ Nâng cấp quyền hạn người dùng

Kịch bản 1: $\sim u u k$

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

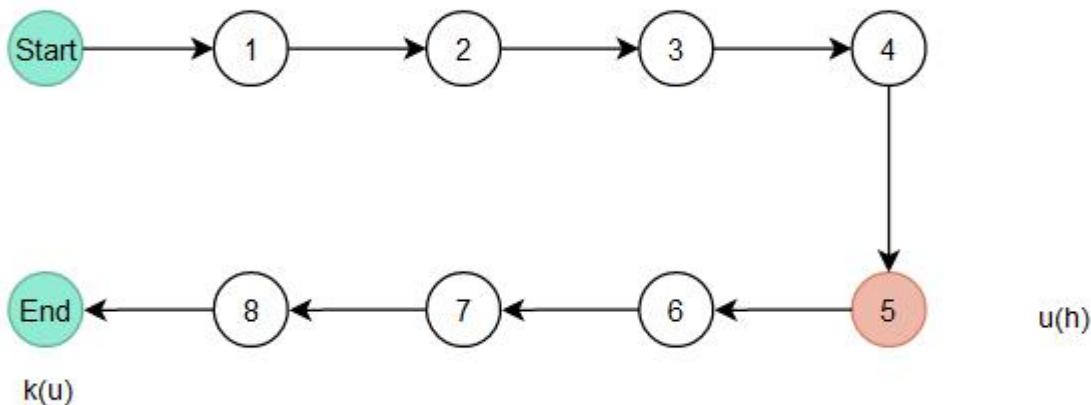
- Kiểm thử đời sóng biển $g(imageId)$:



Hình 6.22.g: Đồ thị đời sóng biển $g(imageId)$ Nâng cấp quyền hạn người dùng
Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

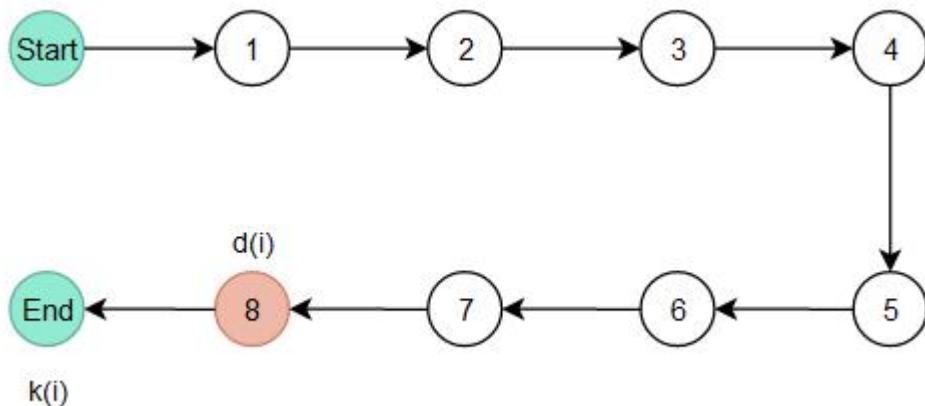
- Kiểm thử đời sóng biển $h/avatar$:



Hình 6.22.h: Đồ thị đời sóng biển $h/avatar$ Nâng cấp quyền hạn người dùng
Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biển chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đòi hỏi biên i (success):

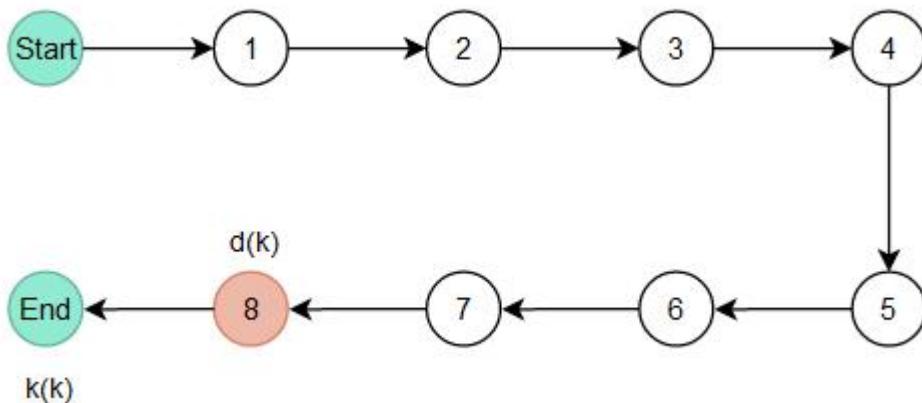


Hình 6.22.i: Đồ thị đòi hỏi biên i (success) Nâng cấp quyền hạn người dùng

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đòi hỏi biên k (message):



Hình 6.22.k: Đồ thị đòi hỏi biên k (message) Nâng cấp quyền hạn người dùng

Kịch bản 1: ~dk

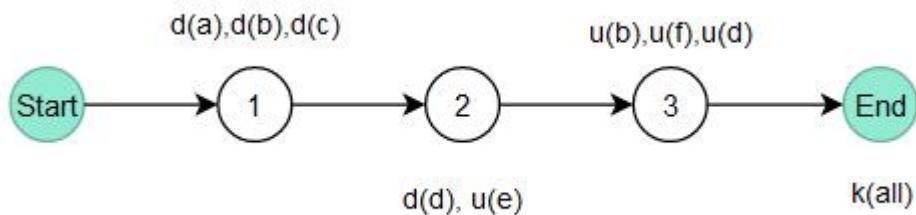
=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

23. Admin - See All Products

```
//Get All Product - Admin Role
exports.getAdminProducts = catchAsyncError(async (req, res, next) => {
  const products = await Product.find(); /*(2)*/

  res.status(200).json({
    success: true,
    products,
  }); /*(3)*/
});
```

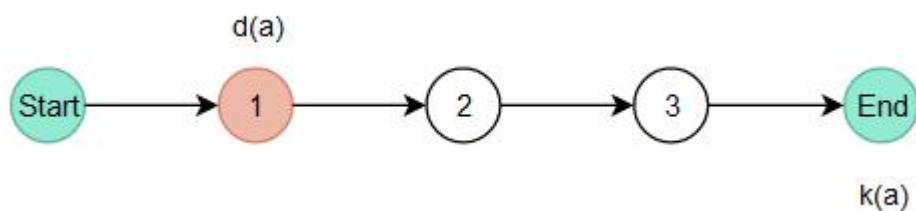
Table 6.23: Code Backend Xem tất cả các sản phẩm



Hình 6.23: Đồ thị dòng dữ liệu Xem tất cả các sản phẩm

Kiểm thử dòng sống 6 biến: a(req), b(res), c(next), d(products), e(Product), f(success)

- Kiểm thử dòng sống biến a(req):

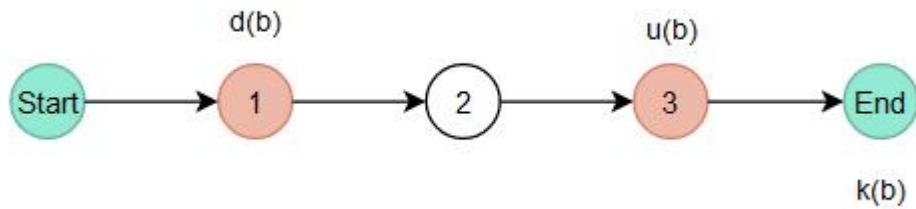


Hình 6.23.a: Đồ thị dòng sống biến a(req) Xem tất cả các sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển $b(res)$:

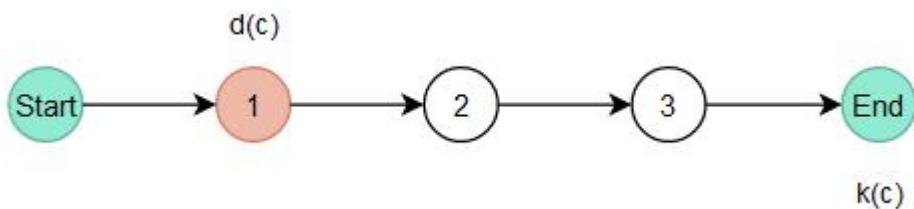


Hình 6.23.b: Đồ thị đời sóng biển $b(res)$ Xem tất cả các sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển $c(next)$:

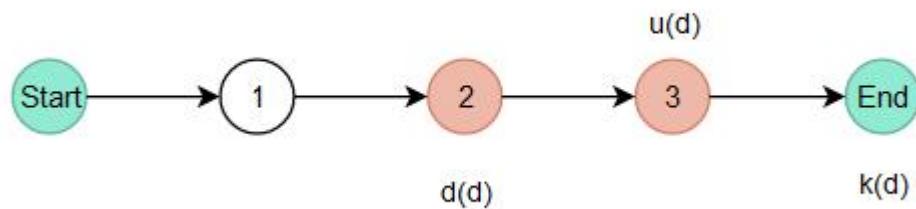


Hình 6.23.c: Đồ thị đời sóng biển $c(next)$ Xem tất cả các sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển $d(products)$:

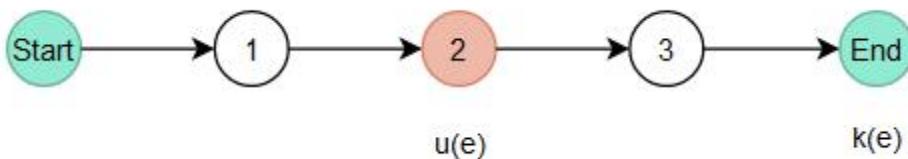


Hình 6.23.d: Đồ thị đời sóng biển $d(products)$ Xem tất cả các sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển e(Product):**

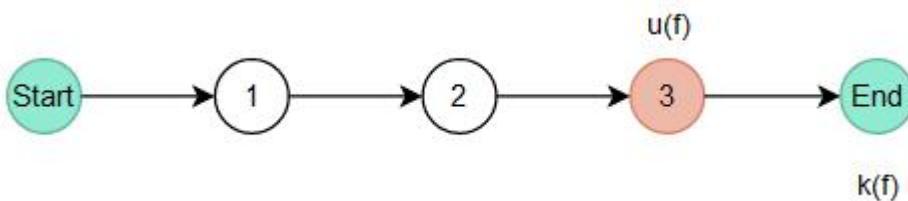


Hình 6.23.e: Đồ thị đời sống biển e(Product) Xem tất cả các sản phẩm

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- **Kiểm thử đời sống biển f(success):**



Hình 6.20.f: Đồ thị đời sống biển f(success) Xem tất cả các sản phẩm

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

24. Admin - Create Products

```
//Create A Product - Admin Role
exports.createProduct = catchAsyncError(async (req, res, next) => {
  let images = [];
  if (typeof req.body.images === "string") {
    images.push(req.body.images);
  } else {
    images = req.body.images;
  }
})
```

```

const imagesLinks = [];

for (let i = 0; i < images.length; i++) /*(6)*/
{
    const result = await cloudinary.v2.uploader.upload(images[i], {
        folder: "products",
    }); /*(8)*/

    imagesLinks.push({
        public_id: result.public_id,
        url: result.secure_url,
    }); /*(9)*/ 
}

req.body.images = imagesLinks; /*(10)*/

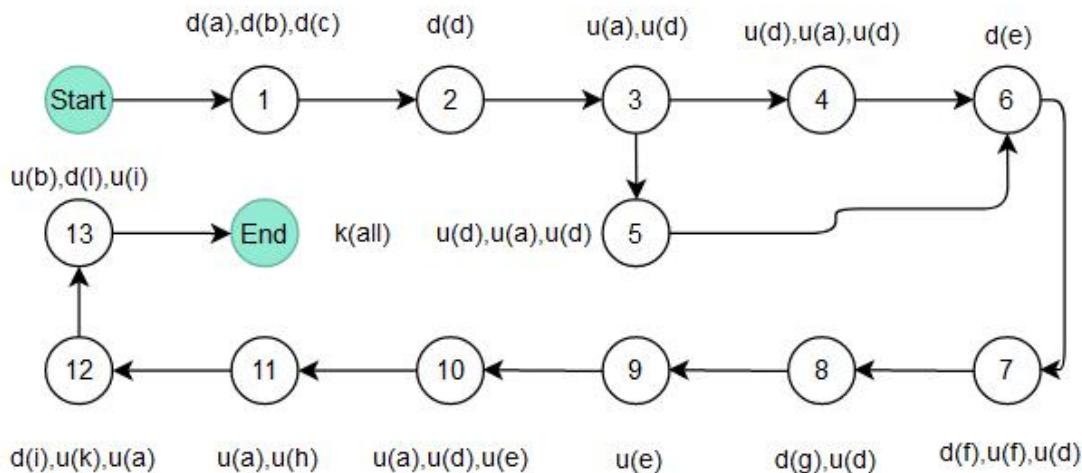
req.body.user = req.user.id; /*(11)*/

const product = await Product.create(req.body); /*(12)*/

res.status(201).json({
    success: true,
    product,
}); /*(13)*/ 
}

```

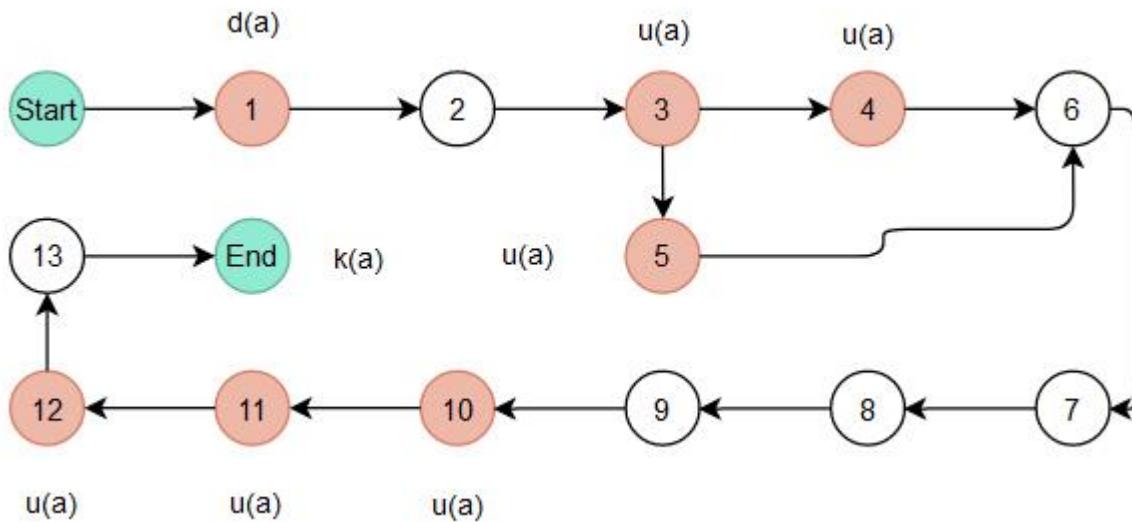
Table 6.24: Code Backend Tạo sản phẩm mới



Hình 6.24: Đồ thị dòng dữ liệu Tạo sản phẩm mới

Kiểm thử dòng dữ liệu 11 biến: a(req), b(res), c(next), d(images), e(imageLinks), f(i), g(result), h(user), i(product), k(Product), l(success)

- Kiểm thử đời sóng biển a(req):



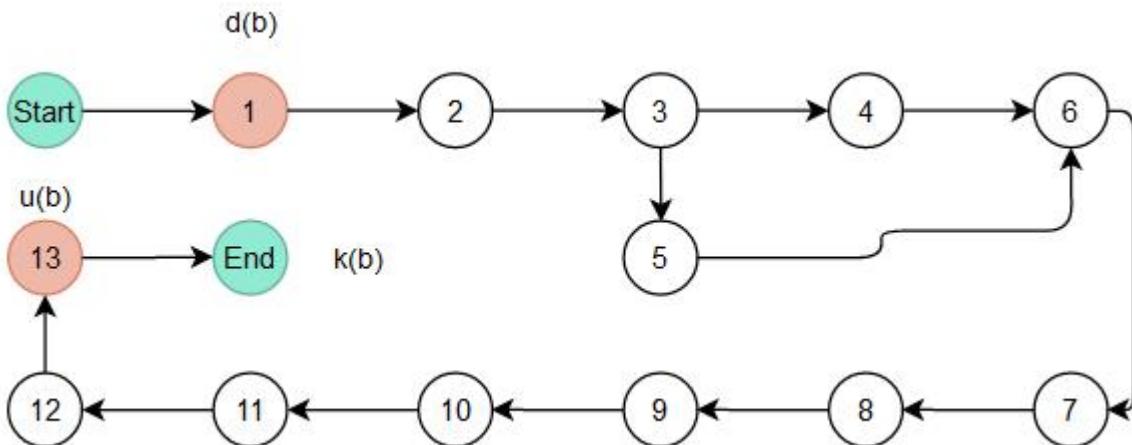
Hình 6.24.a: Đồ thị đời sóng biển a(req) Tạo sản phẩm mới

Kịch bản 1: ~duuuuuuk

Kịch bản 2: ~duuuuuuk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sóng biển b(res):



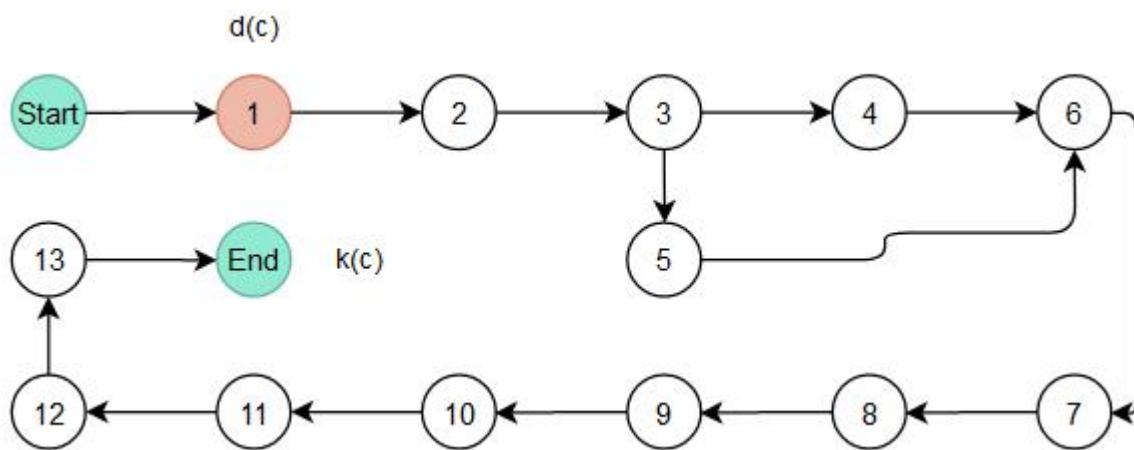
Hình 6.24.b: Đồ thị đời sóng biển b(res) Tạo sản phẩm mới

Kịch bản 1: ~duk

Kịch bản 2: ~duk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đời sống biển c(next):**



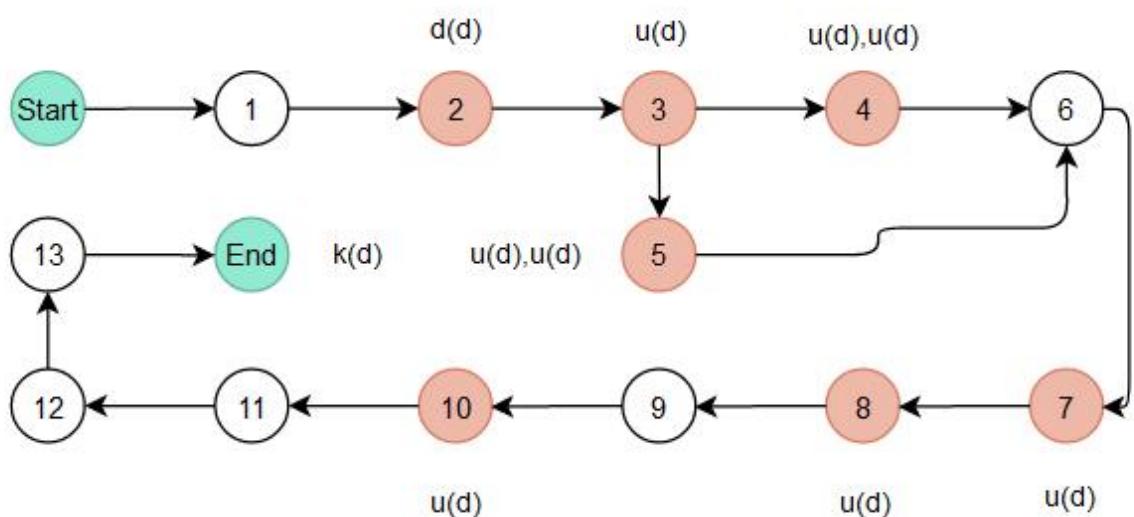
Hình 6.24.c: Đồ thị đời sống biển c(next) Tạo sản phẩm mới

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- **Kiểm thử đời sống biển d(images):**



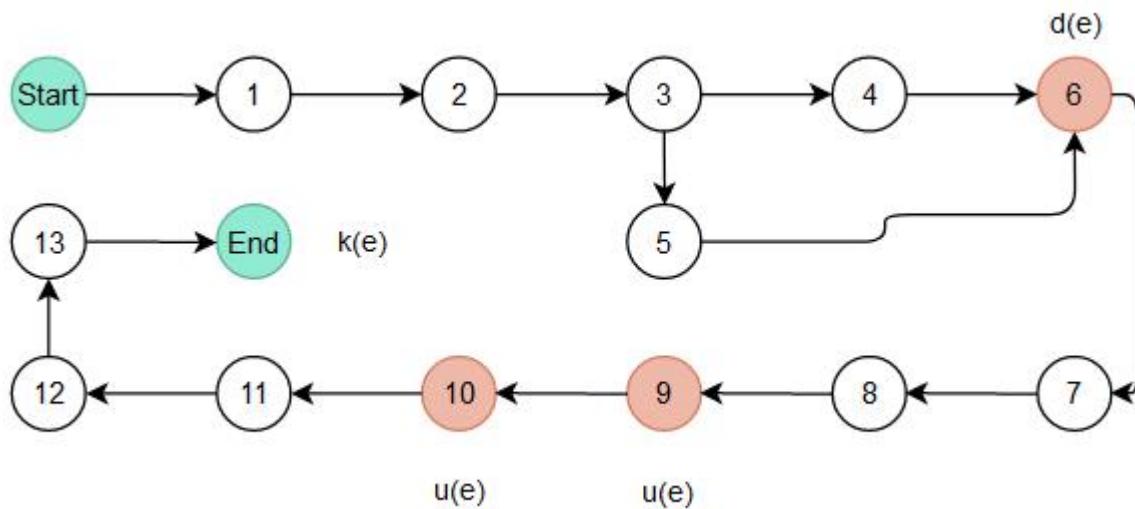
Hình 6.24.d: Đồ thị đời sống biển d(images) Tạo sản phẩm mới

Kịch bản 1: ~duuuuuuk

Kịch bản 2: ~duuuuuuk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đời sóng biển e(imageLinks):**



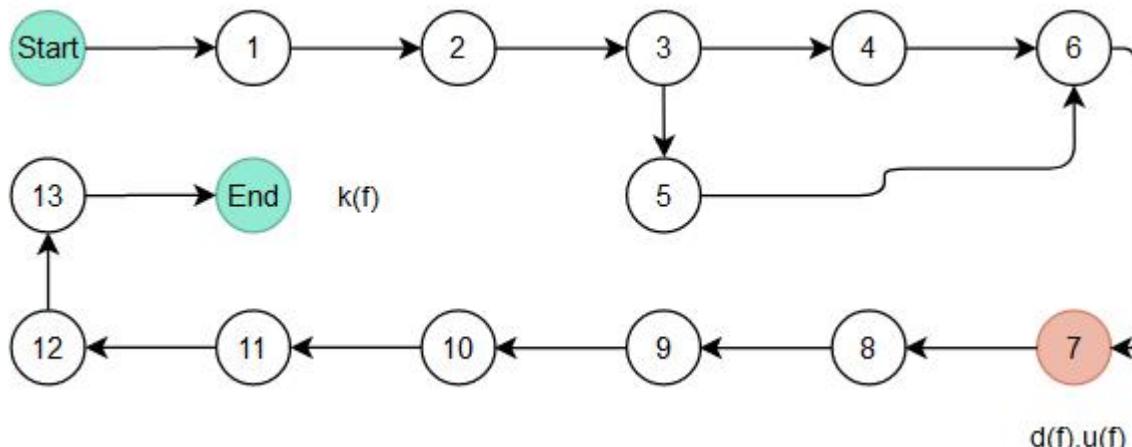
Hình 6.24.e: Đồ thị đời sóng biển e(imageLinks) Tạo sản phẩm mới

Kịch bản 1: ~duuuk

Kịch bản 2: ~duuuk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đời sóng biển f(i):**



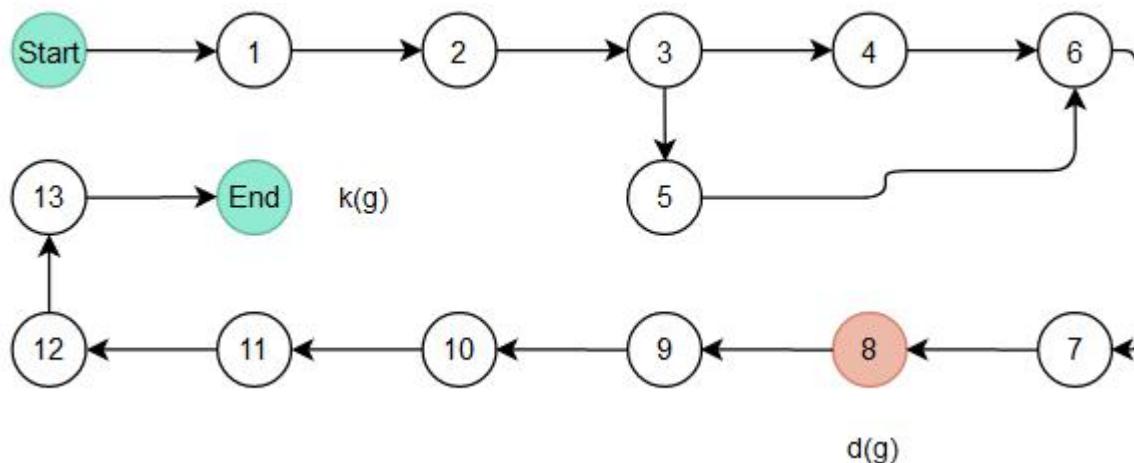
Hình 6.24.f: Đồ thị đòn sóng biến $f(i)$ Tạo sản phẩm mới

Kịch bản 1: ~duk

Kịch bản 2: ~duk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đòn sóng biến $g(result)$:**



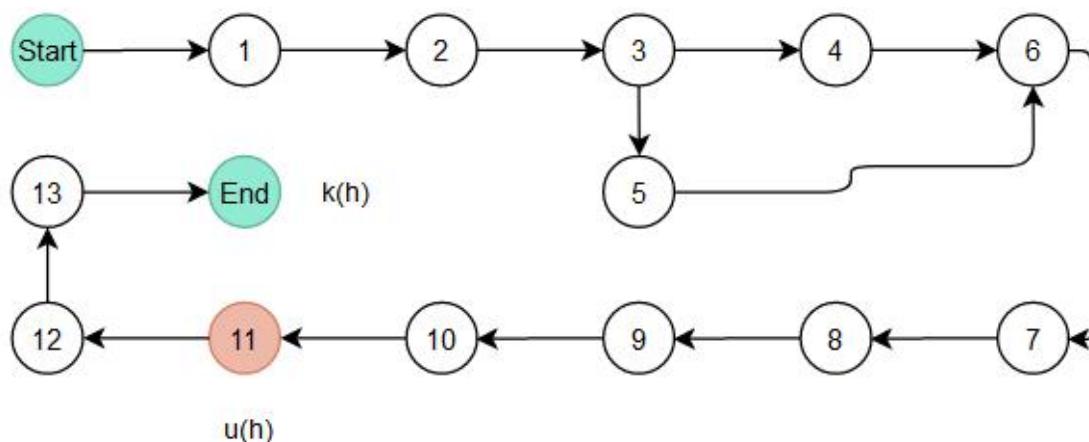
Hình 6.24.g: Đồ thị đòn sóng biến $g(result)$ Tạo sản phẩm mới

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- **Kiểm thử đòn sóng biến $h(user)$:**



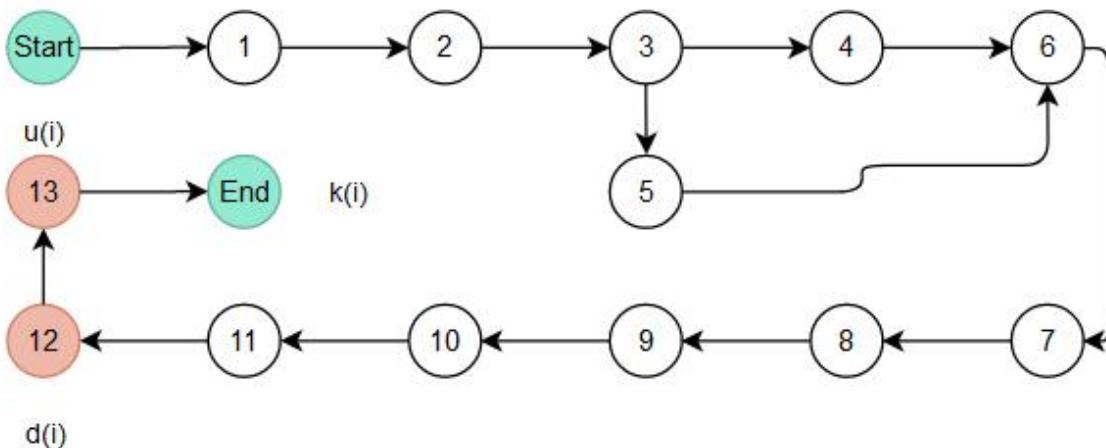
Hình 6.24.h: Đồ thị dòng sống biển $h(\text{user})$ Tạo sản phẩm mới

Kịch bản 1: ~uk

Kịch bản 2: ~uk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biển chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- **Kiểm thử dòng sống biển i (product):**



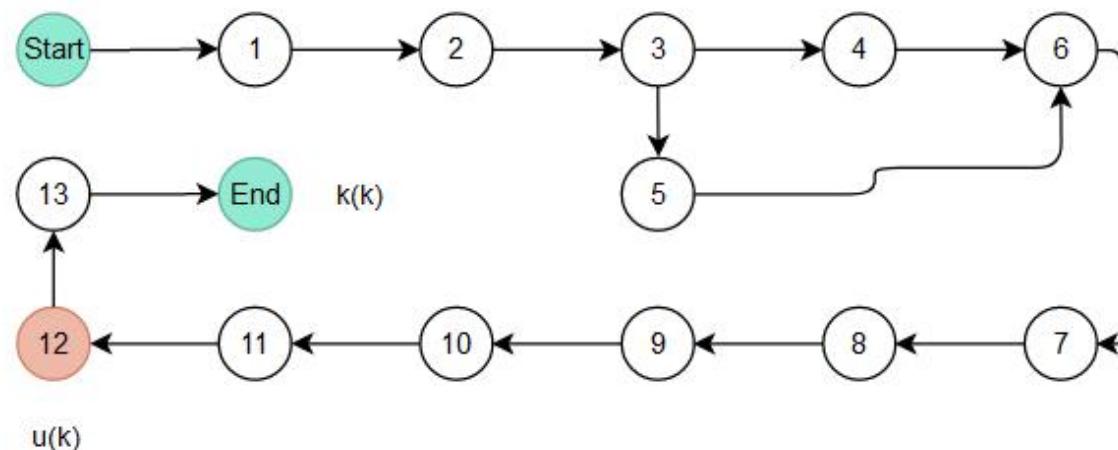
Hình 6.24.i: Đồ thị dòng sống biển i (product) Tạo sản phẩm mới

Kịch bản 1: ~duk

Kịch bản 2: ~duk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử dòng sống biển k (Product):**



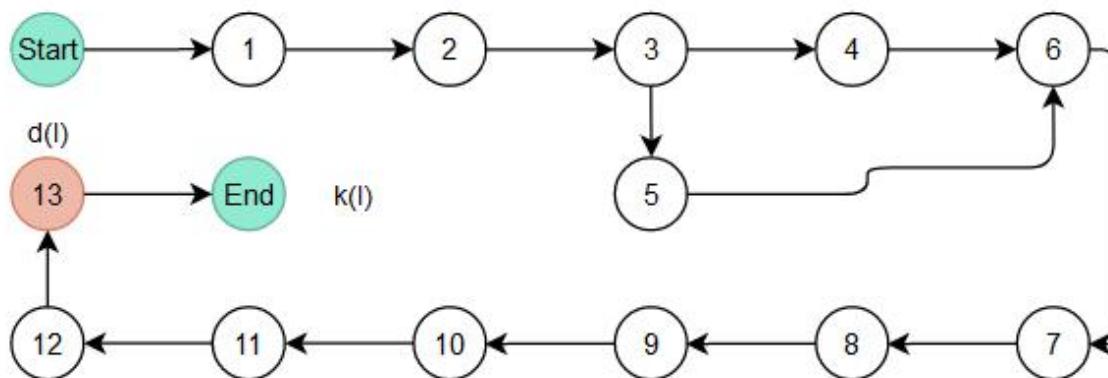
Hình 6.24.k: Đồ thị đòi hỏi biến k(Product) Tạo sản phẩm mới

Kịch bản 1: ~uk

Kịch bản 2: ~uk

=> **Kết luận:** Ở cả 2 kịch bản, việc **~u** miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- **Kiểm thử đòi hỏi biến l(success):**



Hình 6.24.l: Đồ thị đòi hỏi biến l(success) Tạo sản phẩm mới

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

25. Admin - Update Products

```
//Update A Product - Admin Role
exports.updateProduct = catchAsyncError(async (req, res, next) => {
  let product = await Product.findById(req.params.id); /*(2)*/

  if (!product /*(3)*/) {
    return next(new ErrorHandler("Product not found", 404)); /*(4)*/
  }

  // Images Update:
  let images = []; /*(5)*/

  if (typeof req.body.images === "string") /*(6)*/ {
    images.push(req.body.images); /*(7)*/
  }
})
```

```

} else {
    images = req.body.images; /*(8)*/
}

if (images !== undefined /*(9)*/) {
    // Deleting Images From Cloudinary
    for (let i = 0; i < product.images.length; i++) /*(10)*{
        await cloudinary.v2.uploader.destroy(product.images[i].public_id); /*(11)*
    }

    const imagesLinks = []; /*(12)*

    for (let i = 0; i < images.length; i++) /*(13)*{
        const result = await cloudinary.v2.uploader.upload(images[i], {
            folder: "products",
       }); /*(14)*

        imagesLinks.push({
            public_id: result.public_id,
            url: result.secure_url,
       }); /*(15)*
    }

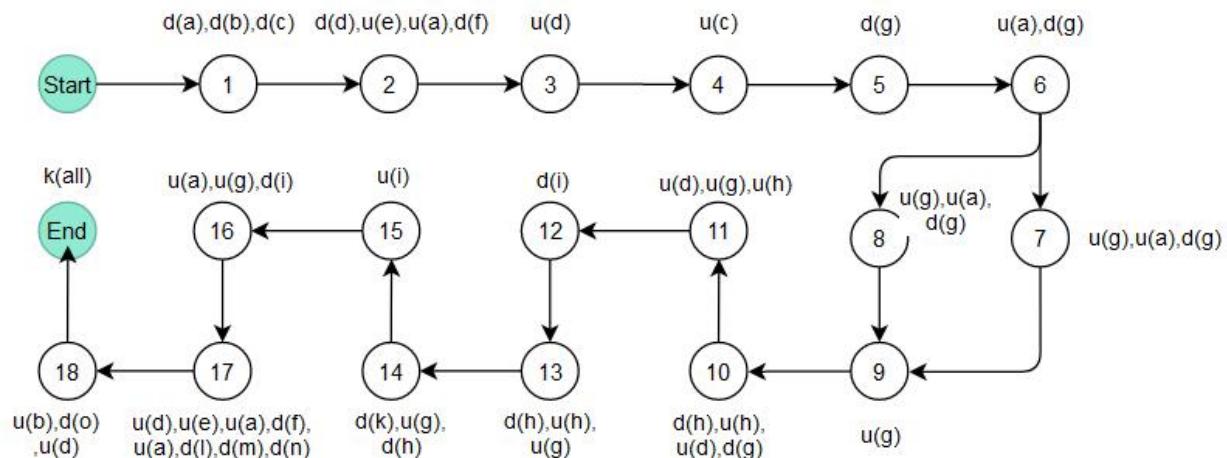
    req.body.images = imagesLinks; /*(16)*
}

product = await Product.findByIdAndUpdate(req.params.id, req.body, {
    new: true,
    runValidators: true,
    useFindAndModify: false,
}); /*(17)*

res.status(200).json({
    success: true,
    product,
}); /*(18)*
}

```

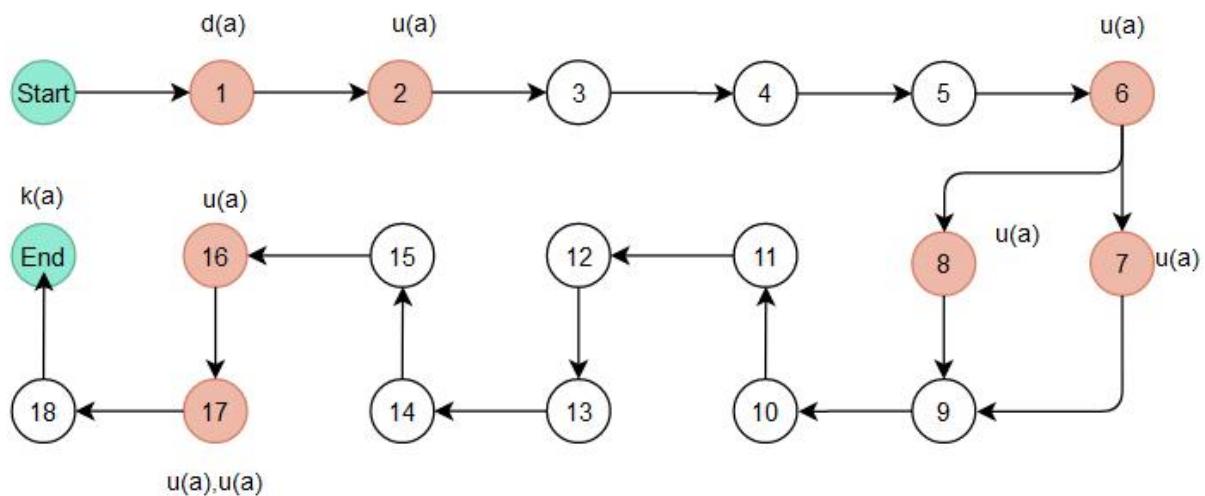
Table 6.25: Code Backend Cập nhật 1 sản phẩm



Hình 6.25: Đồ thị dòng dữ liệu Cập nhật 1 sản phẩm

Kiểm thử đời sống 14 biến: a(req), b(res), c(next), d(product), e(Product), f(id), g(images), h(i), i(imagesLink), k(result), l(new), m(runValidators), n(useFindAndModify), o(success)

- **Kiểm thử đời sống biến a(req):**



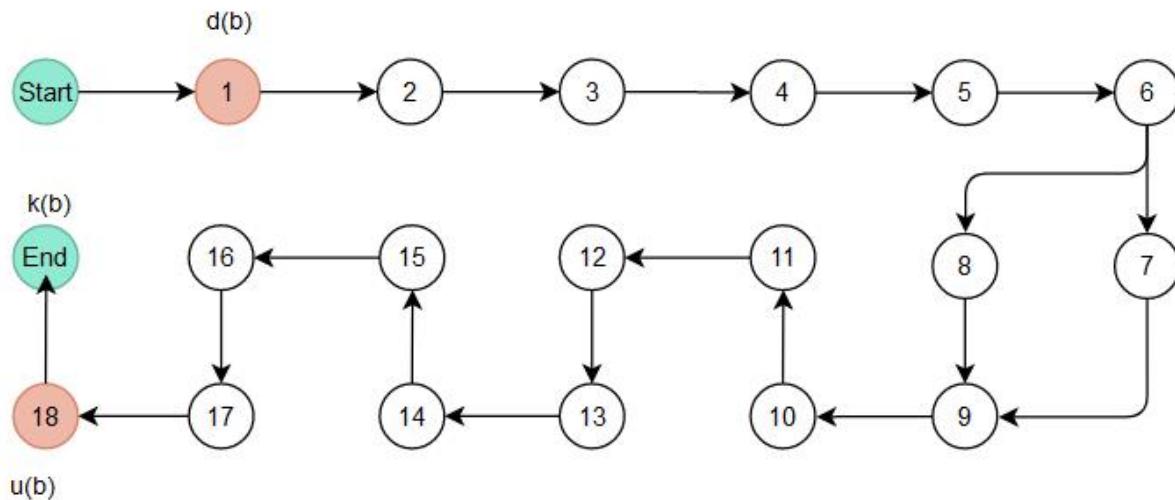
Hình 6.25.a: Đồ thị đời sống biến a(req) Cập nhật 1 sản phẩm

Kịch bản 1: ~duuuuuuk

Kịch bản 2: ~duuuuuuk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sống biến $b(res)$:



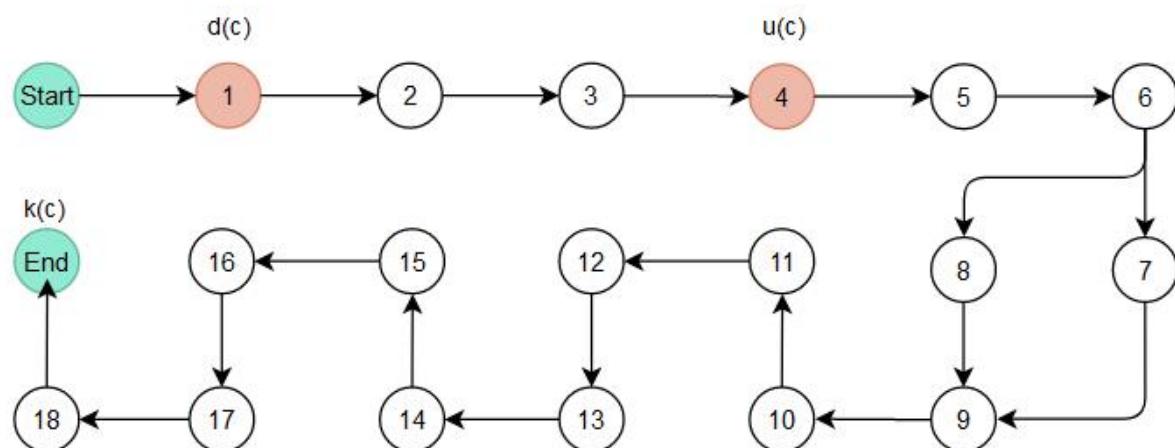
Hình 6.25.b: Đồ thị đời sống biến $b(res)$ Cập nhật 1 sản phẩm

Kịch bản 1: ~duk

Kịch bản 2: ~duk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sống biến $c(next)$:



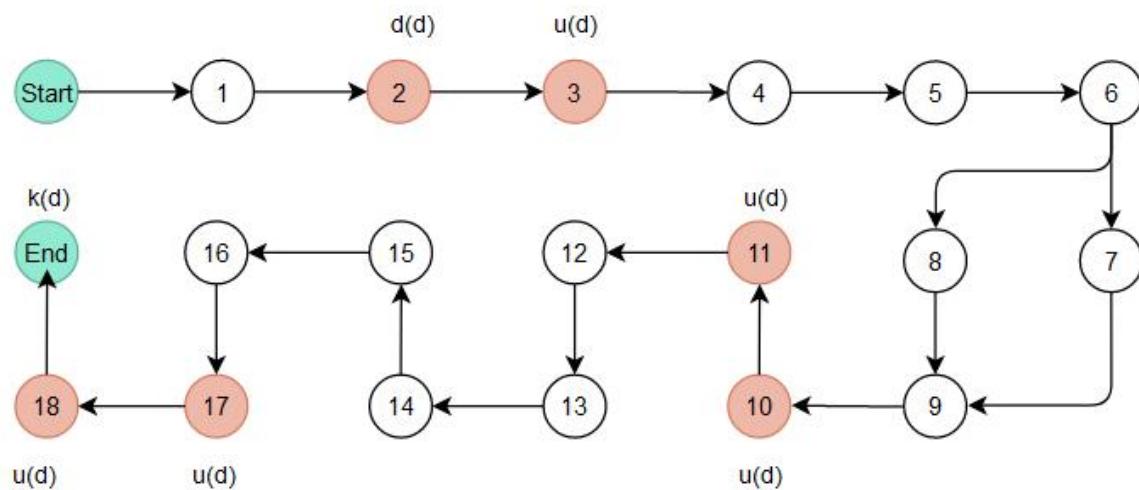
Hình 6.25.c: Đồ thị đời sống biến $c(next)$ Cập nhật 1 sản phẩm

Kịch bản 1: ~duk

Kịch bản 2: ~duk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đời sóng biên d(product):**



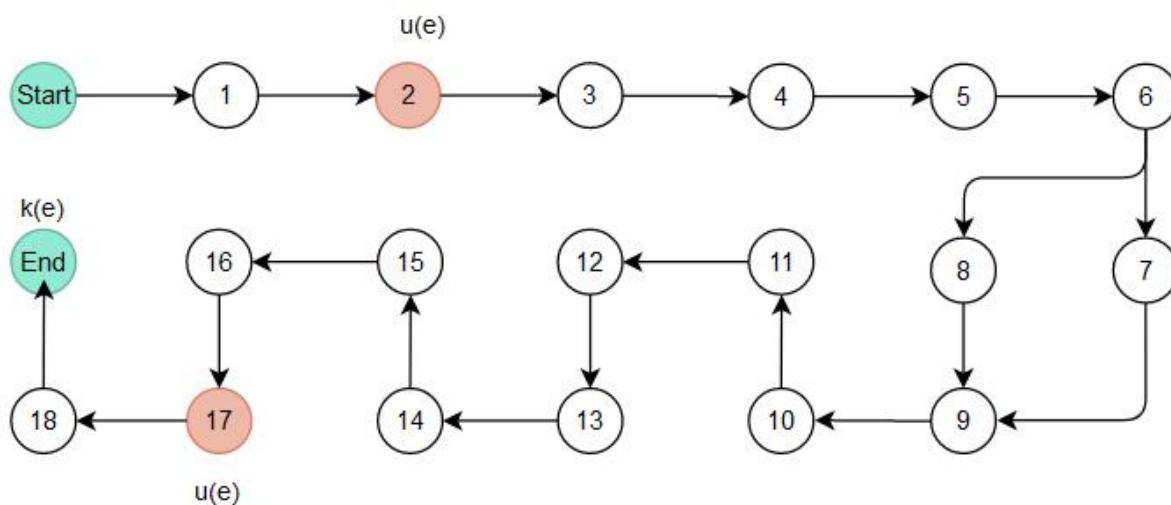
Hình 6.25.d: Đồ thị đời sóng biên d(product) Cập nhật 1 sản phẩm

Kịch bản 1: ~duuuuuk

Kịch bản 2: ~duuuuuuk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đời sóng biên e(Product):**



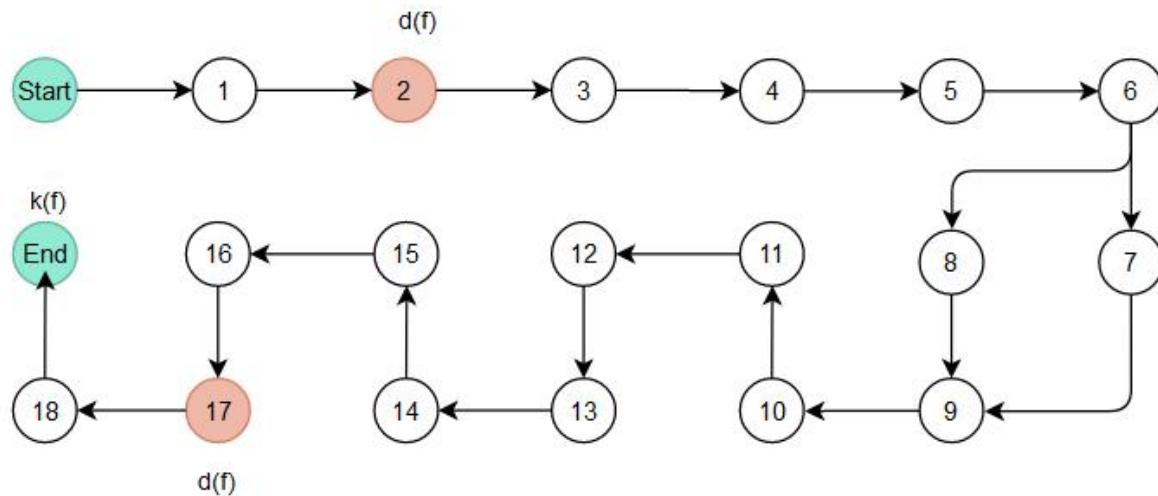
Hình 6.25.e: Đồ thị đời sóng biên e(Product) Cập nhật 1 sản phẩm

Kịch bản 1: ~uuk

Kịch bản 2: ~uuuk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- **Kiểm thử dòi sóng biển f(id):**



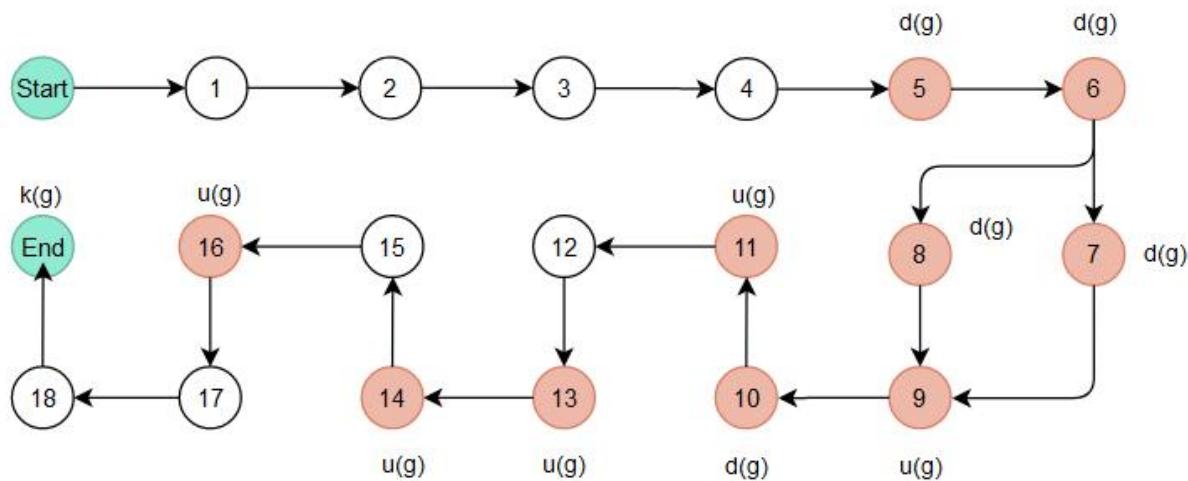
Hình 6.25.f: Đồ thị dòi sóng biển f(id) Cập nhật 1 sản phẩm

Kịch bản 1: ~ddk

Kịch bản 2: ~ddk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dd** biến được định nghĩa rồi định nghĩa nữa, hơi lạ, có thể đúng và chấp nhận được, nhưng cũng có thể có lỗi lập trình; cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển $g(images)$:



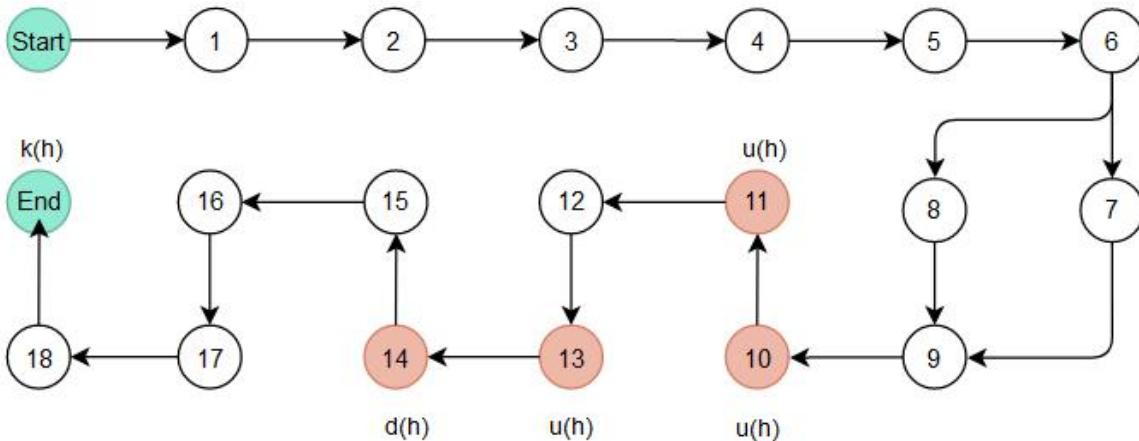
Hình 6.25.g: Đồ thị đời sóng biển $g(images)$ Cập nhật 1 sản phẩm

Kịch bản 1: ~ddduduuuuk

Kịch bản 2: ~ddduduuuuk

=> **Kết luận:** Ở cả 2 kịch bản, các cặp đôi ***dd*** biên được định nghĩa rồi định nghĩa nữa, hơi lạ, có thể đúng và chấp nhận được, nhưng cũng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển $h(i)$:



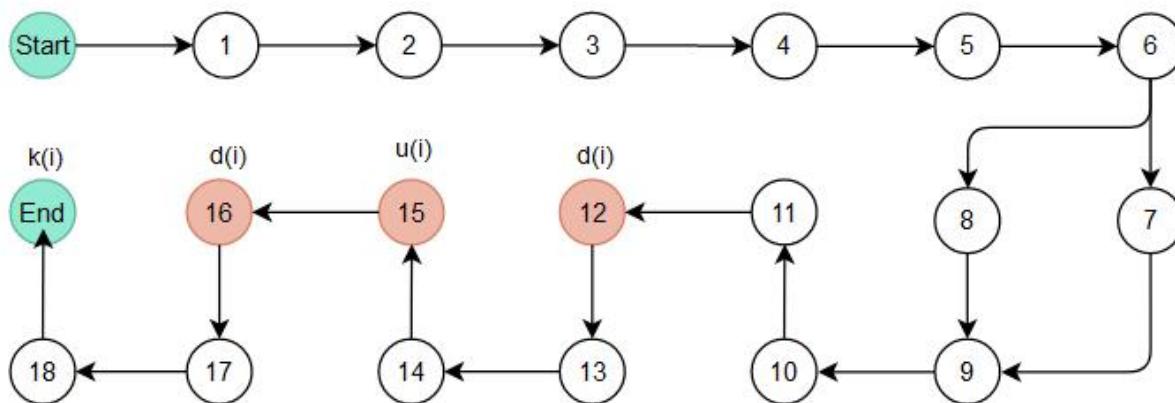
Hình 6.25.h: Đồ thị đời sóng biển $h(i)$ Cập nhật 1 sản phẩm

Kịch bản 1: ~uuudk

Kịch bản 2: ~uuudk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào); cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình; còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biển i(imagesLink):



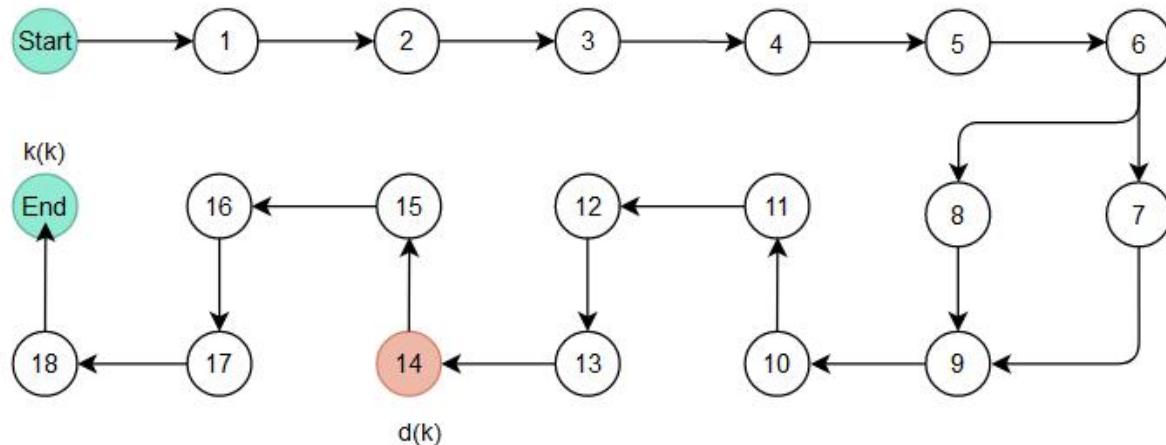
Hình 6.25.i: Đồ thi đời sống biển i(imagesLink) Cập nhật 1 sản phẩm

Kích bản 1: ~dudk

Kích bản 2: ~dudk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lâ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển $k(result)$:



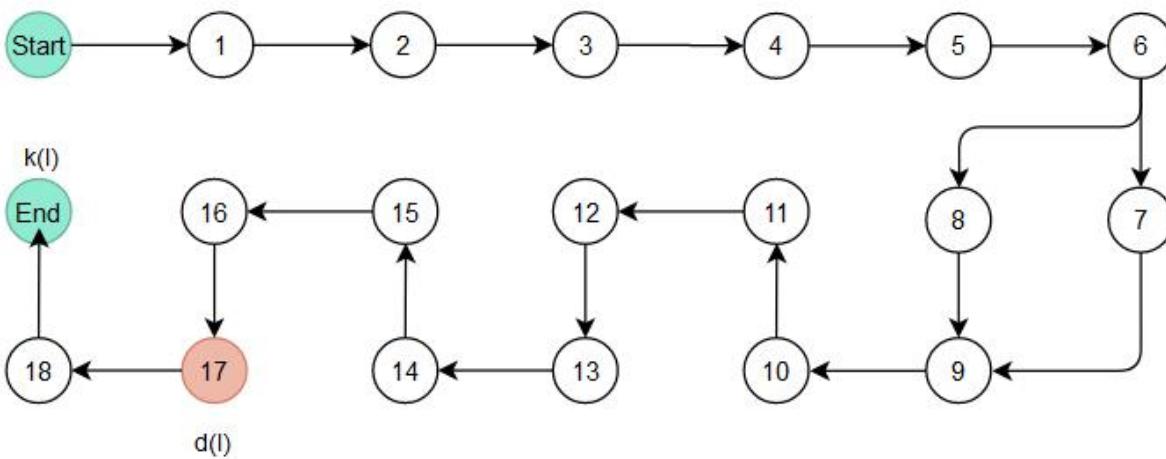
Hình 6.25.k: Đồ thị đời sóng biển $k(result)$ Cập nhật 1 sản phẩm

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển $l(new)$:



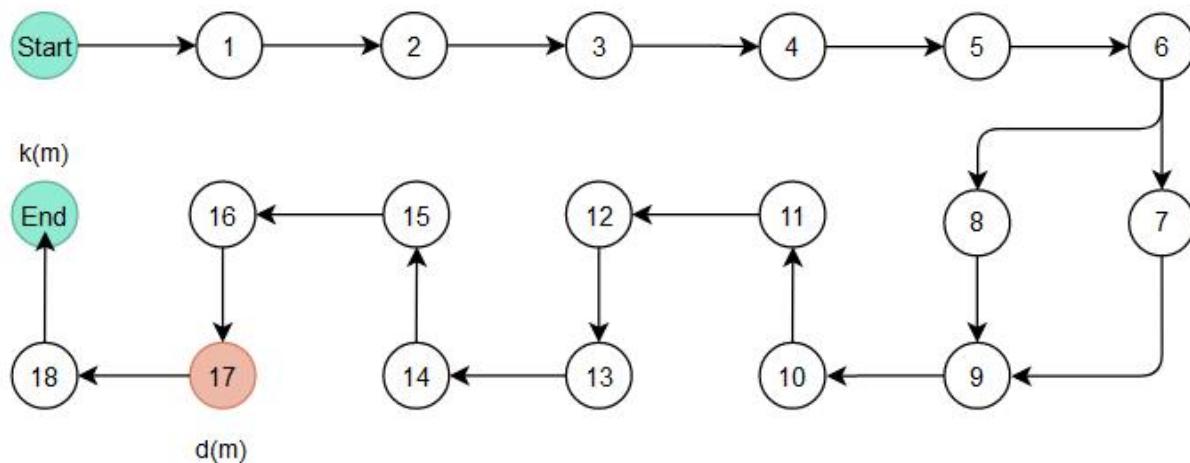
Hình 6.25.l: Đồ thị đời sóng biển $l(new)$ Cập nhật 1 sản phẩm

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biến m(*runValidators*):



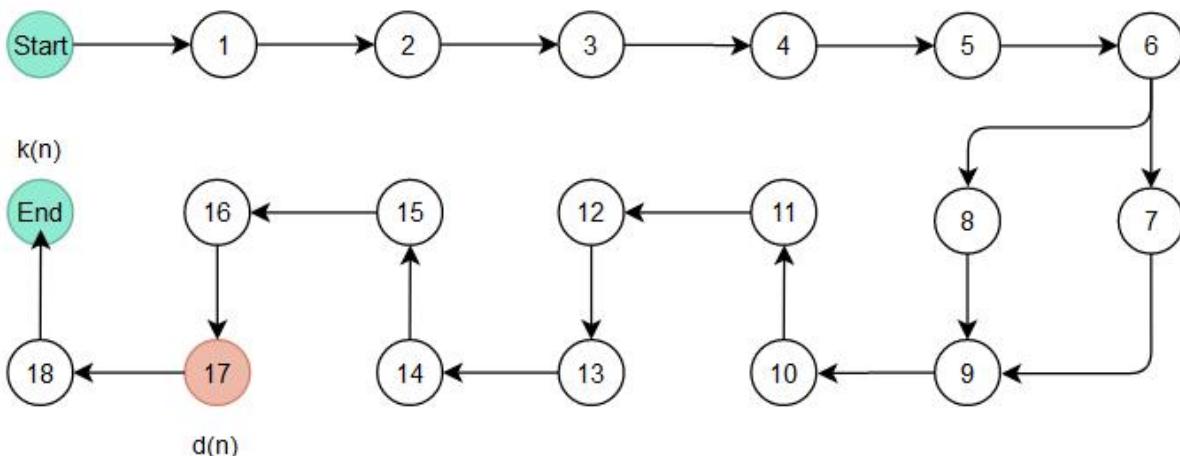
Hình 6.25.m: Đồ thị đời sóng biến m(*runValidators*) Cập nhật 1 sản phẩm

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biến n(*useFindAndModify*):



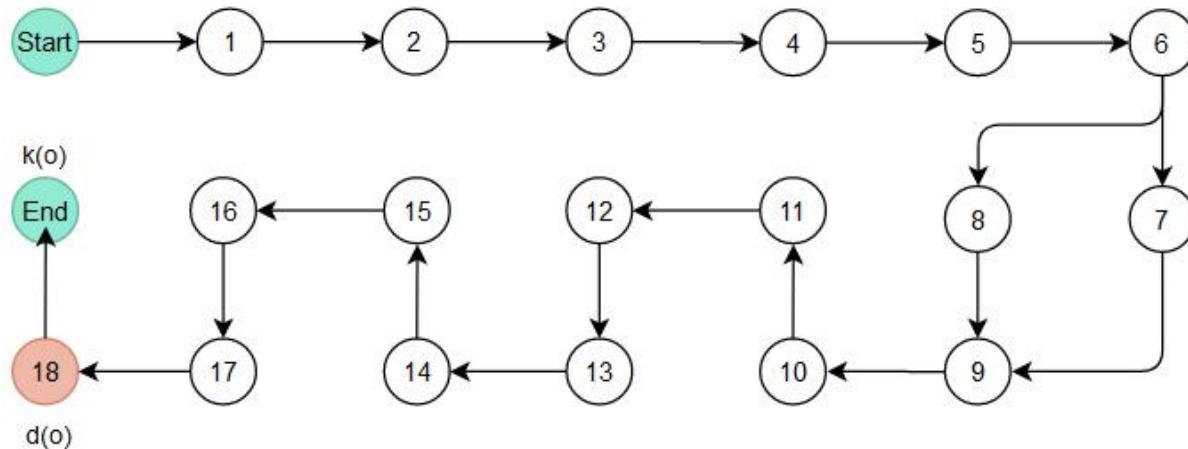
Hình 6.25.n: Đồ thị đời sóng biến n(*useFindAndModify*) Cập nhật 1 sản phẩm

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- **Kiểm thử đời sống biển o(success):**



Hình 6.25.o: Đồ thị đời sống biển o(success) Cập nhật 1 sản phẩm

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

26. Admin - Delete Products

```
//Delete A Product - Admin Role
exports.deleteProduct = catchAsyncError(async (req, res, next/*(1)*)) => {
  const product = await Product.findById(req.params.id); /*(2)*

  if (!product /*(3)*)) {
    return res.status(500).json({
      success: false,
      message: "Product not found",
   }); /*(4)*
  }

  // Deleting Images From Cloudinary
  for (let i = 0; i < product.images.length; i++) /*(5)* {
    await cloudinary.v2.uploader.destroy(product.images[i].public_id); /*(6)*
  }
}
```

```

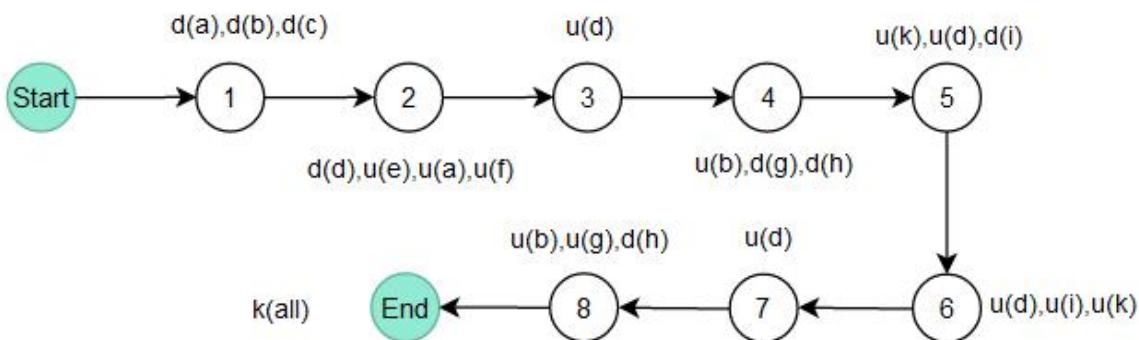
}

await product.remove(); /*(7)*

res.status(200).json({
    success: true,
    message: "Product Delete Successfully",
}); /*(8)*
}

```

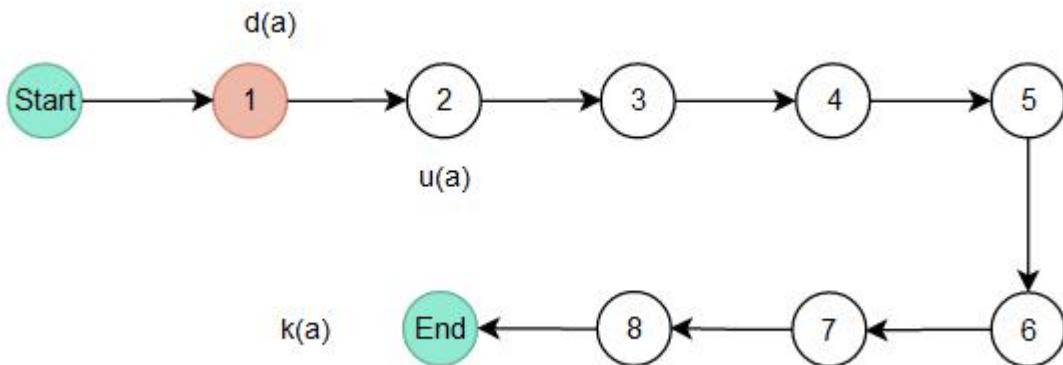
Table 6.26: Code Backend Xóa sản phẩm



Hình 6.26: Đồ thị dòng dữ liệu Xóa sản phẩm

Kiểm thử dòng 10 biến: a(req), b(res), c(next), d(product), e(Product), f(id), g(success), h(message), i(images), k(i)

- Kiểm thử dòng biến a(req):

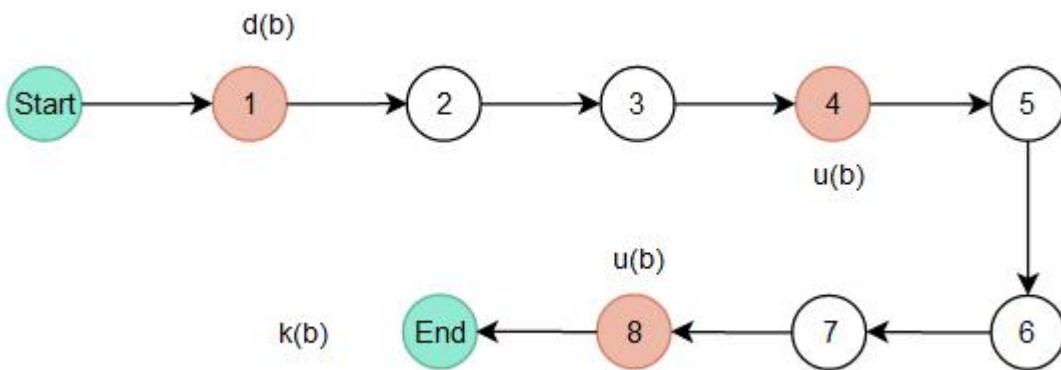


Hình 6.26.a: Đồ thị dòng biến a(req) Xóa sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển b(res):**

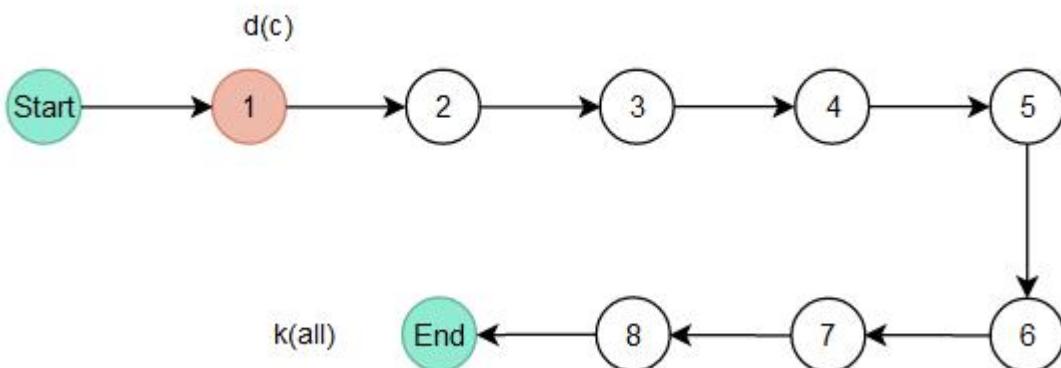


Hình 6.26.b: Đồ thị đời sống biển b(res) Xóa sản phẩm

Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biển c(next):**

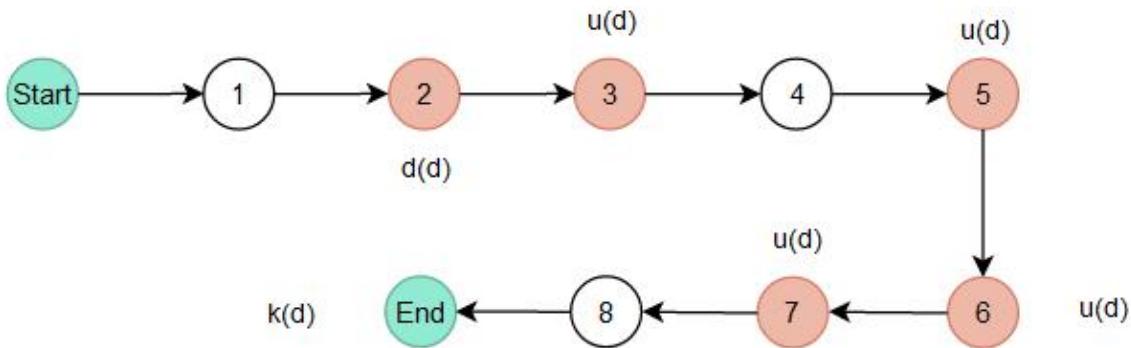


Hình 6.26.c: Đồ thị đời sống biển c(next) Xóa sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển d(product):

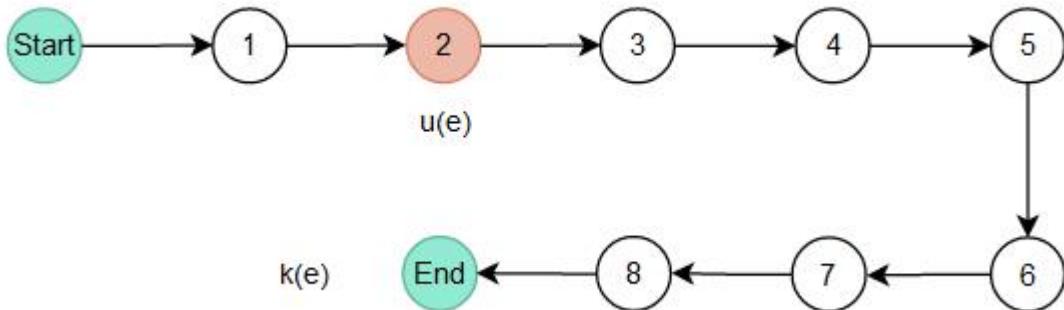


Hình 6.26.d: Đồ thị đời sóng biển d(product) Xóa sản phẩm

Kịch bản 1: ~duuuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển e(Product):

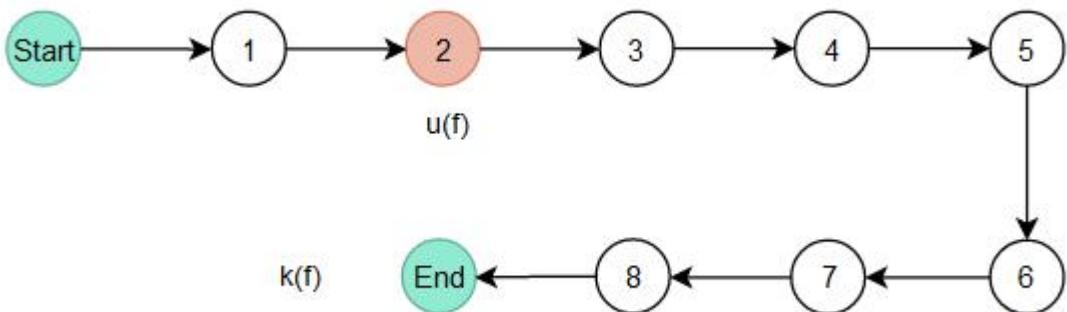


Hình 6.26.e: Đồ thị đời sóng biển e(Product) Xóa sản phẩm

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biển $f(id)$:

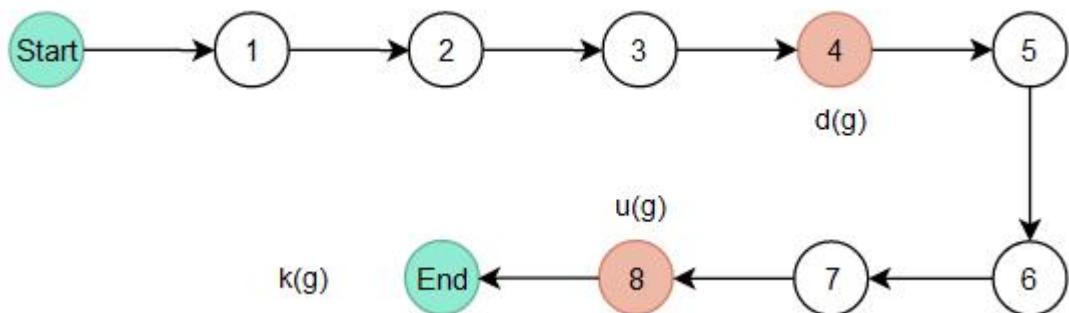


Hình 6.26.f: Đồ thị đời sóng biển $f(id)$ Xóa sản phẩm

Kịch bản 1: $\sim u k$

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biển $g(success)$:

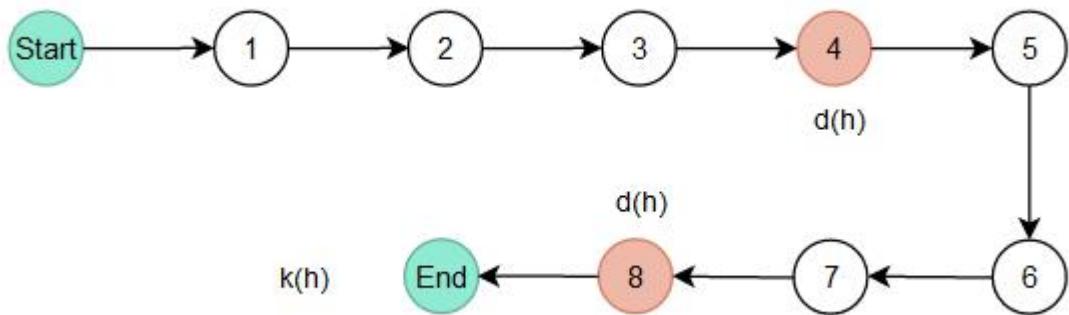


Hình 6.26.g: Đồ thị đời sóng biển $g(success)$ Xóa sản phẩm

Kịch bản 1: $\sim d u k$

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến h (message):

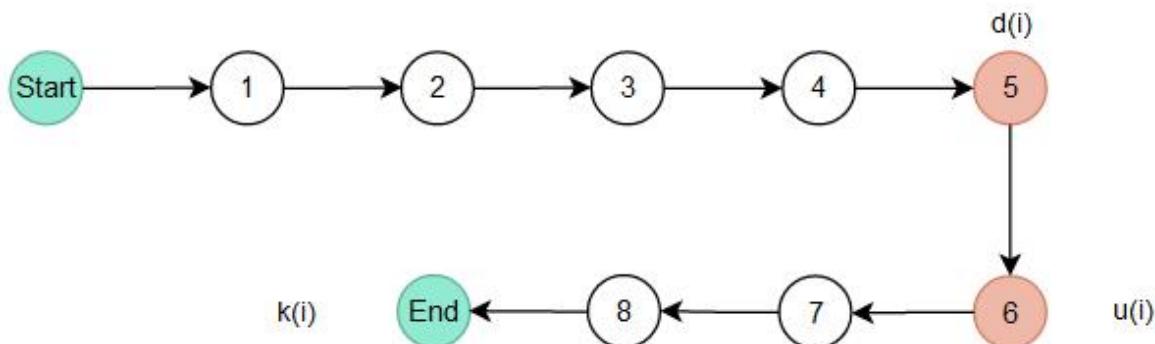


Hình 6.26.h: Đồ thị đời sống biến h (message) Xóa sản phẩm

Kịch bản 1: ~ddk

=> Kết luận: Ở kịch bản 1, cặp đôi **dd** biến được định nghĩa rồi định nghĩa nữa, hơi lả, có thể đúng và chấp nhận được, nhưng cũng có thể có lỗi lập trình; cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lả, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến i (images):

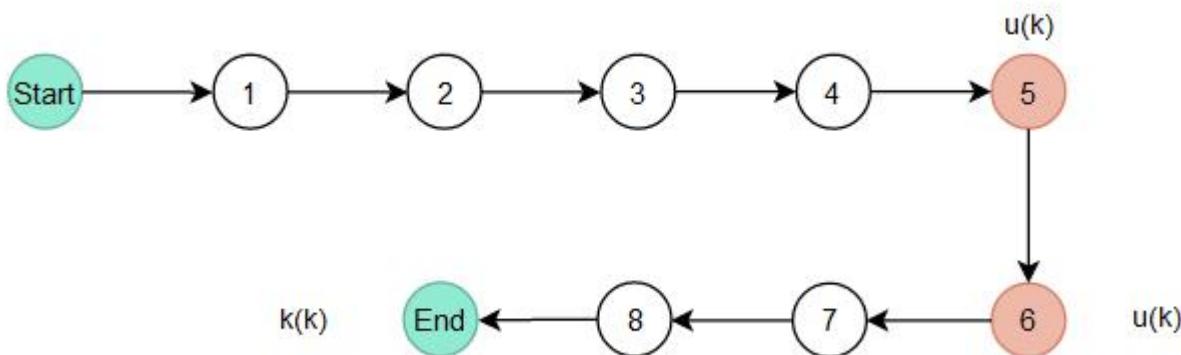


Hình 6.26.i: Đồ thị đời sống biến i (images) Xóa sản phẩm

Kịch bản 1: ~duk

=> Kết luận: Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biển $k(i)$:



Hình 6.26.k: Đồ thị đời sống biển $k(i)$ Xóa sản phẩm

Kịch bản 1: ~uuuk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biển chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

27. Admin - See All Orders on system

```

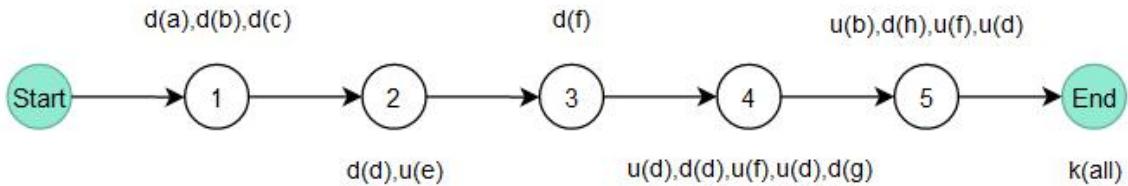
//Get all Orders - Admin Role
exports.getAllOrders = catchAsyncError(async (req, res, next/*(1)*)) => {
  const orders = await Order.find(); /*(2)*

  let totalAmount = 0; /*(3)*

  orders.forEach((order) => {
    totalAmount += order.totalPrice;
 }); /*(4)*

  res.status(200).json({
    success: true,
    totalAmount,
    orders,
 }); /*(5)*
});
  
```

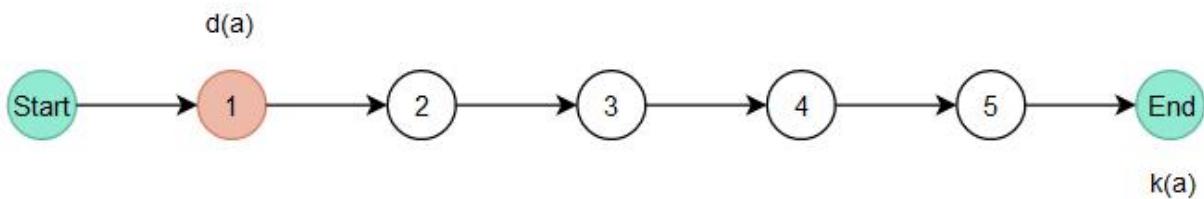
Table 6.27: Code Backend Xem tất cả các đơn hàng trên hệ thống



Hình 6.27: Đồ thị dòng dữ liệu Xem tất cả các đơn hàng trên hệ thống

Kiểm thử dòng sóng 8 biến: $a(req)$, $b(res)$, $c(next)$, $d(orders)$, $e(Order)$, $f(totalAmount)$, $g(totalPrice)$, $h(success)$

- **Kiểm thử dòng sóng biến $a(req)$:**

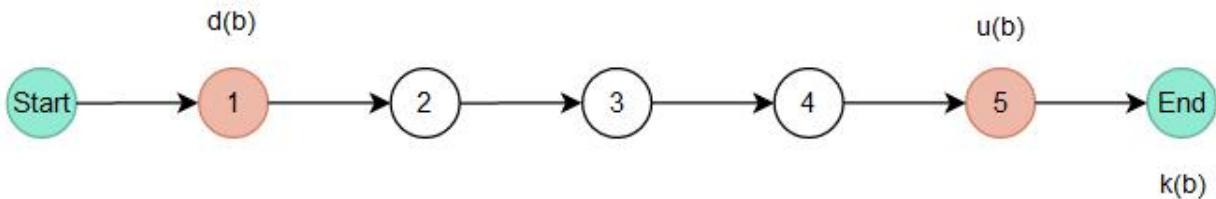


Hình 6.27.a: Đồ thị dòng sóng biến $a(req)$ Xem tất cả các đơn hàng trên hệ thống

Kịch bản 1: $\sim dk$

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- **Kiểm thử dòng sóng biến $b(res)$:**

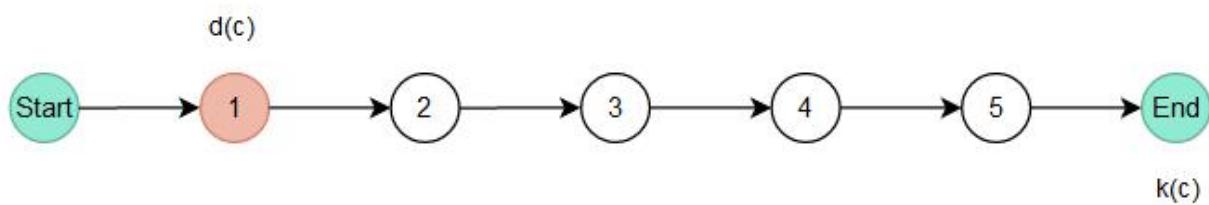


Hình 6.27.b: Đồ thị dòng sóng biến $b(res)$ Xem tất cả các đơn hàng trên hệ thống

Kịch bản 1: $\sim duk$

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biến $c(next)$:

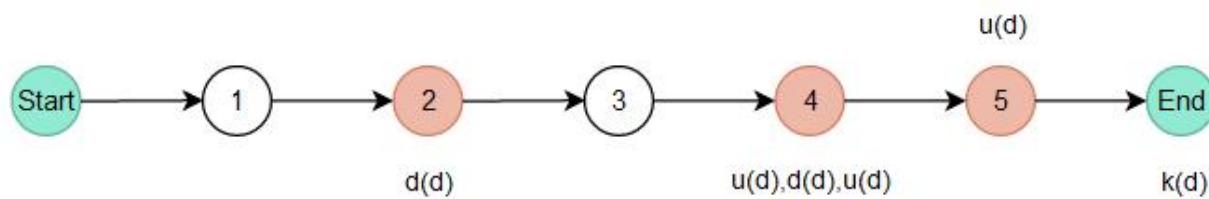


Hình 6.27.c: Đồ thị đời sóng biến $c(next)$ Xem tất cả các đơn hàng trên hệ thống

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biến $d(orders)$:

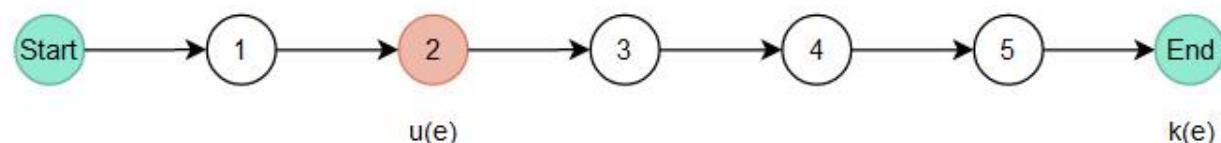


Hình 6.27.d: Đồ thị đời sóng biến $d(orders)$ Xem tất cả các đơn hàng trên hệ thống

Kịch bản 1: ~duduuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biến $e(Order)$:

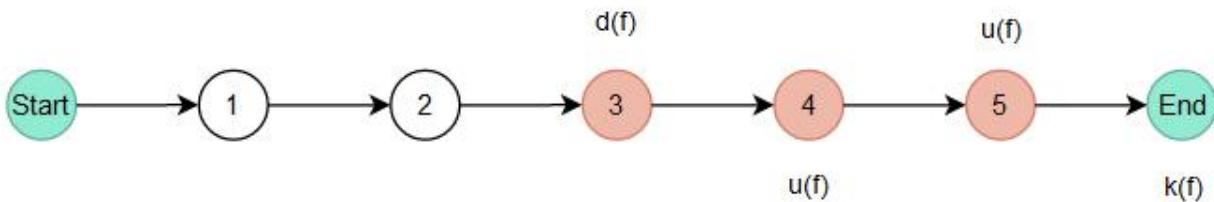


Hình 6.27.e: Đồ thị đời sóng biến $e(Order)$ Xem tất cả các đơn hàng trên hệ thống

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

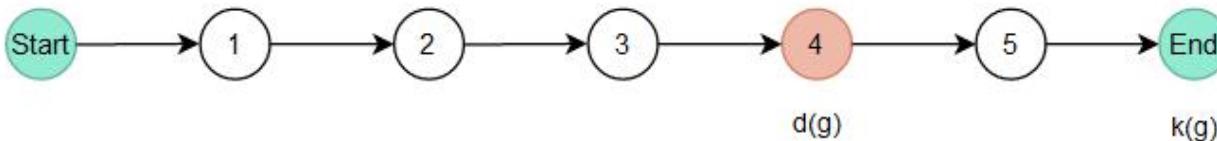
- Kiểm thử đời sóng biển $f(totalAmount)$:



Hình 6.27.f: Đồ thị đời sóng biển $f(totalAmount)$ Xem tất cả các đơn hàng trên hệ thống
Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

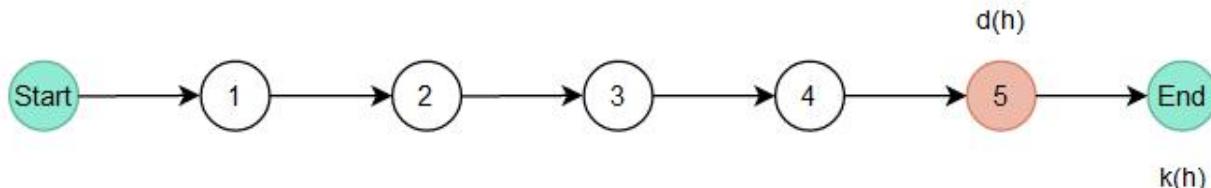
- Kiểm thử đời sóng biển $g(totalPrice)$:



Hình 6.27.g: Đồ thị đời sóng biển $g(totalPrice)$ Xem tất cả các đơn hàng trên hệ thống
Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển $h(success)$:



Hình 6.27.h: Đồ thị đời sóng biển $h(success)$ Xem tất cả các đơn hàng trên hệ thống
Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

28. Admin - Update An Order

```
//Update Order Status - Admin Role
exports.updateOrder = catchAsyncError(async (req, res, next) => {
  const order = await Order.findById(req.params.id); /*(2)*

  if (!order /*(3)*) {
    return next(new ErrorHander("Order not found with this Id", 404)); /*(4)*
  }

  if (order.orderStatus === "Delivered" /*(5)*) {
    return next(new ErrorHander("You have already delivered this order", 400)); /*(6)*
  }

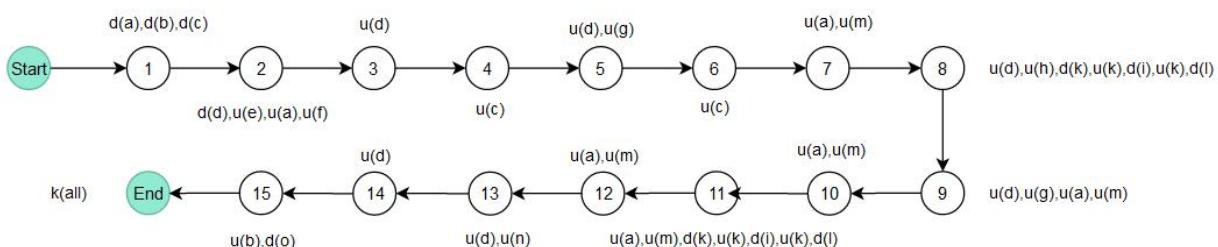
  if (req.body.status === "Shipped" /*(7)*) {
    order.orderItems.forEach(async (o) => {
      await updateStock(o.product, o.quantity);
    }); /*(8)*
  }
  order.orderStatus = req.body.status; /*(9)*

  if (req.body.status === "Shipped" /*(10)*) {
    order.orderItems.forEach(async (o) => {
      await updateStock(o.product, o.quantity);
    }); /*(11)*
  }

  if (req.body.status === "Delivered" /*(12)*) {
    order.deliveredAt = Date.now(); /*(13)*
  }

  await order.save({ validateBeforeSave: false }); /*(14)*
  res.status(200).json({
    success: true,
 }); /*(15)*
});
```

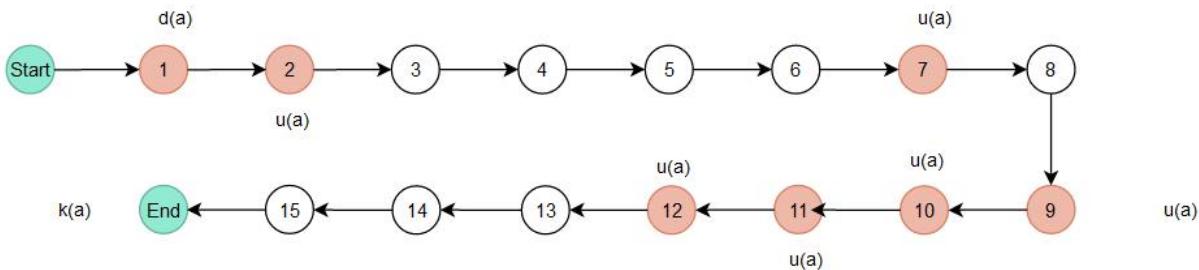
Table 6.28: Code Backend Cập nhật trạng thái đơn hàng



Hình 6.28: Đồ thị dòng dữ liệu Cập nhật trạng thái đơn hàng

Kiểm thử đời sống 14 biến: $a(req)$, $b(res)$, $c(next)$, $d(order)$, $e(Order)$, $f(id)$,
 $g(orderStatus)$, $h(orderItems)$, $i(product)$, $k(o)$, $l(quantity)$, $m(status)$, $n(deliveredAt)$,
 $o(success)$

- **Kiểm thử đời sống biến $a(req)$:**

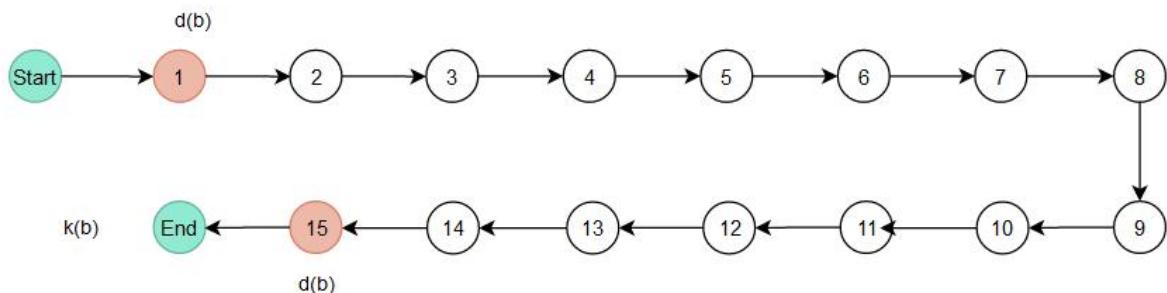


Hình 6.28.a: Đồ thị đời sống biến $a(req)$ Cập nhật trạng thái đơn hàng

Kịch bản 1: ~duuuuuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử đời sống biến $b(res)$:**

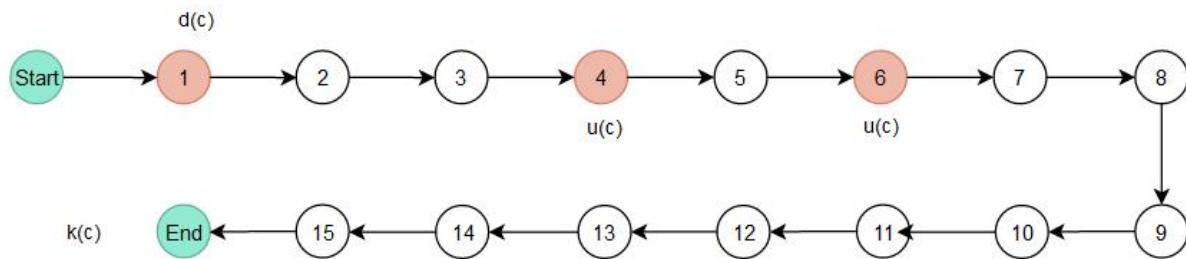


Hình 6.28.b: Đồ thị đời sống biến $b(res)$ Cập nhật trạng thái đơn hàng

Kịch bản 1: ~ddk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dd biến được định nghĩa rồi định nghĩa nữa, hơi lả, có thể đúng và chấp nhận được, nhưng cũng có thể có lỗi lập trình; cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lả, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sóng biển $c(next)$:

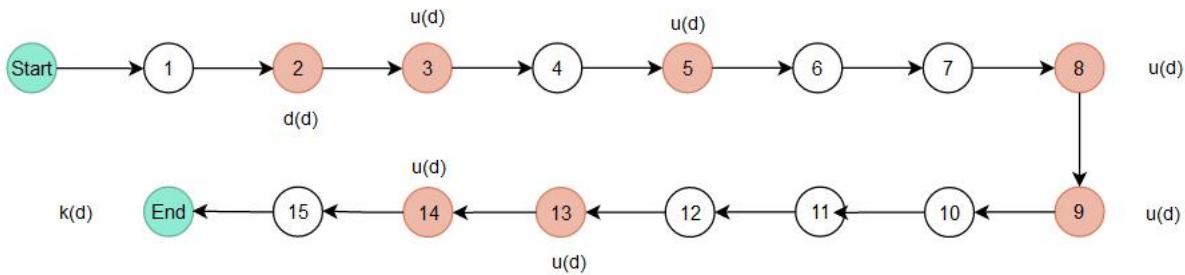


Hình 6.28.c: Đồ thị đời sóng biển $c(next)$ Cập nhật trạng thái đơn hàng

Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển $d(order)$:

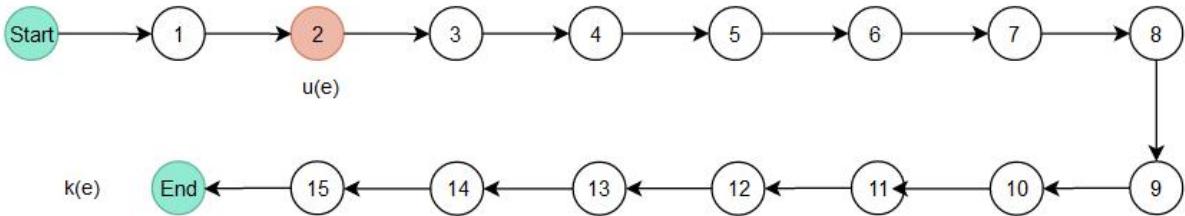


Hình 6.28.d: Đồ thị đời sóng biển $d(order)$ Cập nhật trạng thái đơn hàng

Kịch bản 1: ~duuuuuuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biển $e(Order)$:

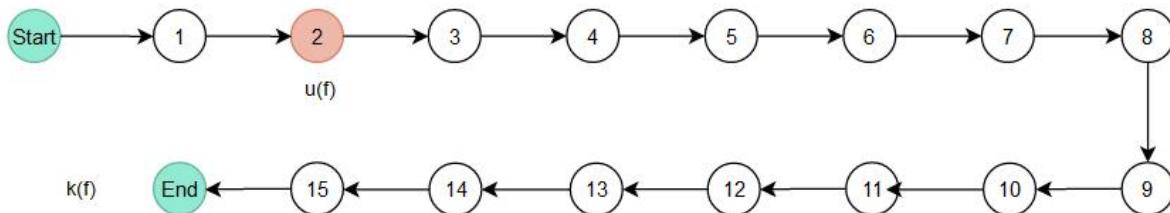


Hình 6.28.e: Đồ thị đời sóng biển $e(Order)$ Cập nhật trạng thái đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biến $f(id)$:

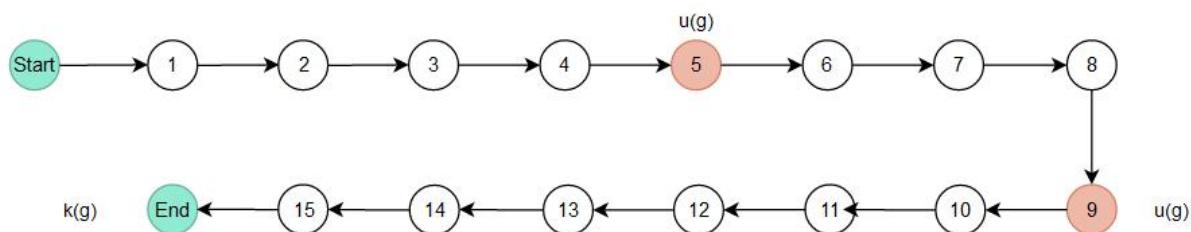


Hình 6.28.f: Đồ thị đời sống biến $f(id)$ Cập nhật trạng thái đơn hàng

Kịch bản 1: $\sim uuk$

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biến $g(orderStatus)$:

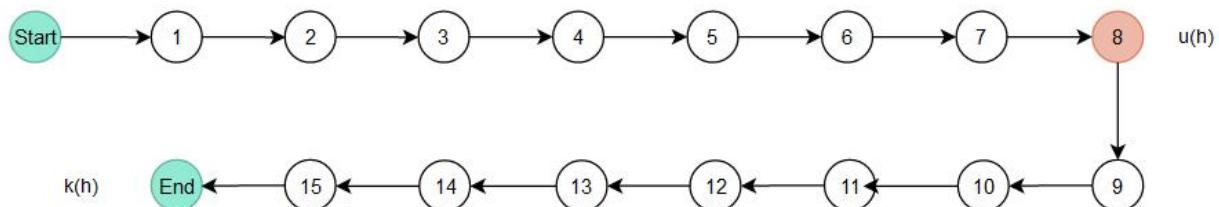


Hình 6.28.g: Đồ thị đời sống biến $g(orderStatus)$ Cập nhật trạng thái đơn hàng

Kịch bản 1: $\sim uuuk$

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biến $h(orderItems)$:

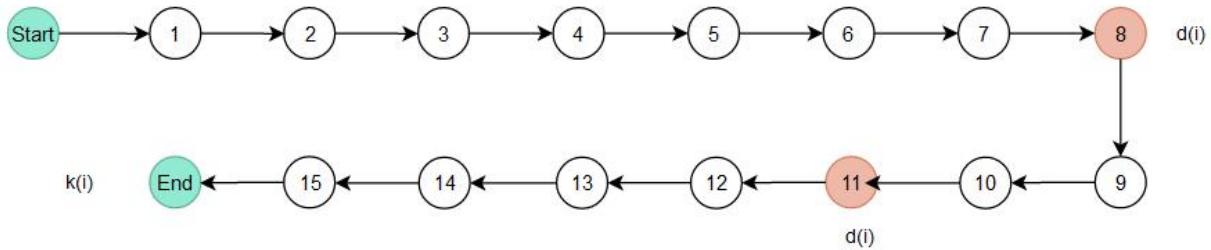


Hình 6.28.h: Đồ thị đòn súng biến h (orderItems) Cập nhật trạng thái đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đòn súng biến i (product):

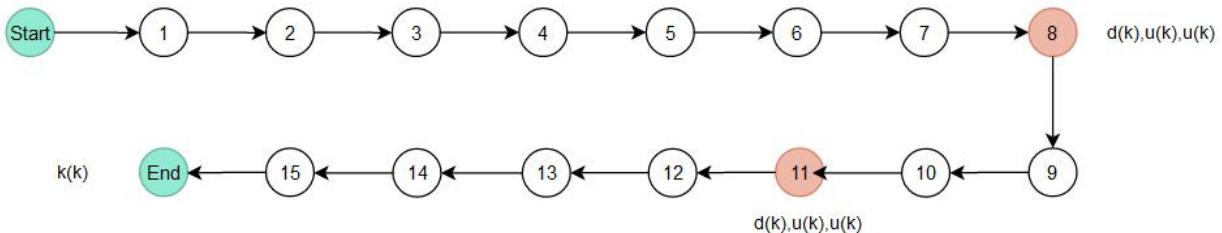


Hình 6.28.i: Đồ thị đòn súng biến i (product) Cập nhật trạng thái đơn hàng

Kịch bản 1: ~ddk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dd biến được định nghĩa rồi định nghĩa nữa, hơi lạt, có thể đúng và chấp nhận được, nhưng cũng có thể có lỗi lập trình; cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạt, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đòn súng biến $k(o)$:

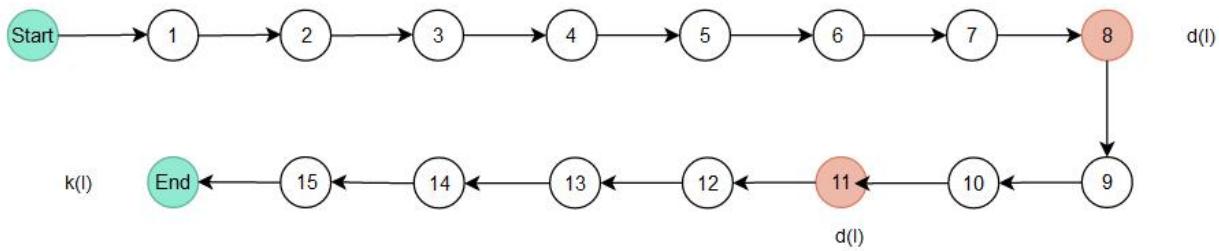


Hình 6.28.k: Đồ thị đòn súng biến $k(o)$ Cập nhật trạng thái đơn hàng

Kịch bản 1: ~duuduuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sống biến l(quantity):

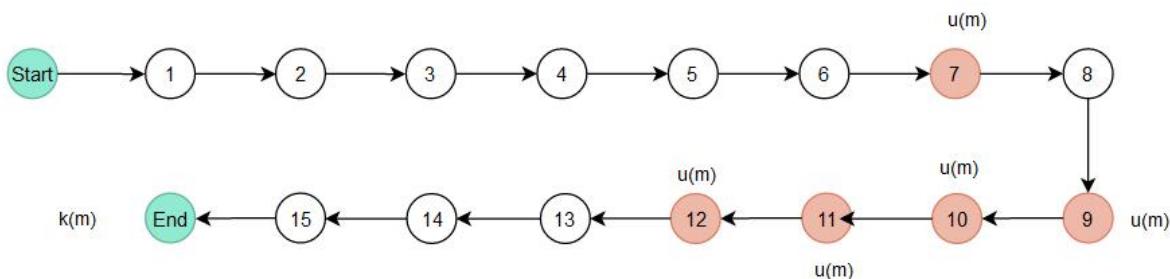


Hình 6.28.l: Đồ thị đời sống biến l(quantity) Cập nhật trạng thái đơn hàng

Kịch bản 1: ~ddk

=> **Kết luận:** Ở kịch bản 1, cặp đôi **dd** biến được định nghĩa rồi định nghĩa nữa, hơi lả, có thể đúng và chấp nhận được, nhưng cũng có thể có lỗi lập trình; cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lả, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến m(status):

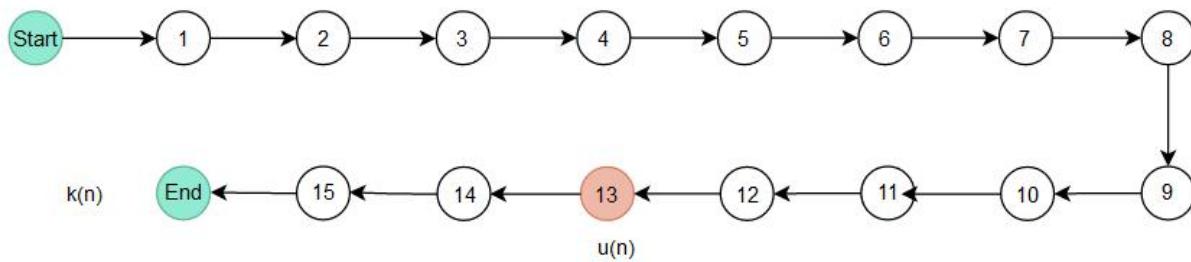


Hình 6.28.m: Đồ thị đời sống biến m(status) Cập nhật trạng thái đơn hàng

Kịch bản 1: ~uuuuuk

=> **Kết luận:** Ở kịch bản 1, việc **~u** miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biến n (deliveredAt):

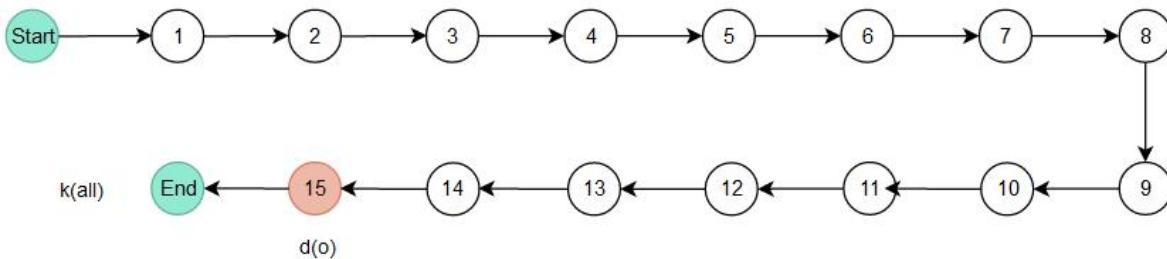


Hình 6.28.n: Đồ thị đời sống biến n (deliveredAt) Cập nhật trạng thái đơn hàng

Kịch bản 1: $\sim u k$

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biến o (success):



Hình 6.28.o: Đồ thị đời sống biến o (success) Cập nhật trạng thái đơn hàng

Kịch bản 1: $\sim dk$

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

29. Admin - Delete An Order

```
//Delete Order - Admin Role
exports.deleteOrder = catchAsyncError(async (req, res, next /*(1)*/) => {
  const order = await Order.findById(req.params.id); /*(2)*/
  if (!order /*(3)*/) {
    return next(new ErrorHandler("Order not found with this Id", 404)); /*(4)*/
  }
})
```

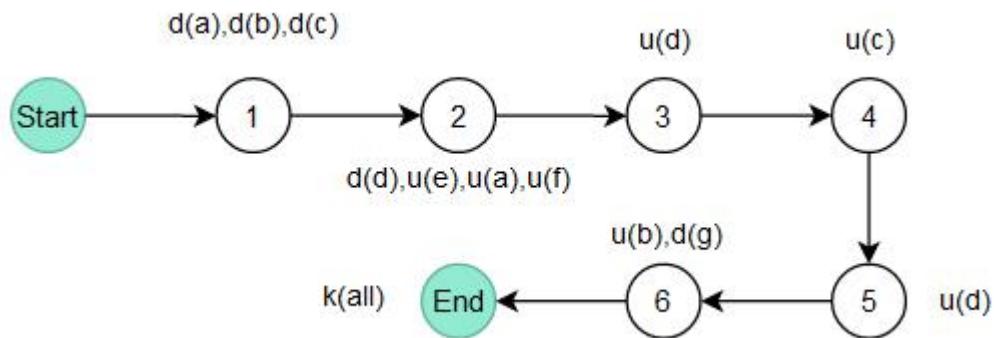
```

await order.remove(); /*(5)*/

res.status(200).json({
  success: true,
}); /*(6)*/
);

```

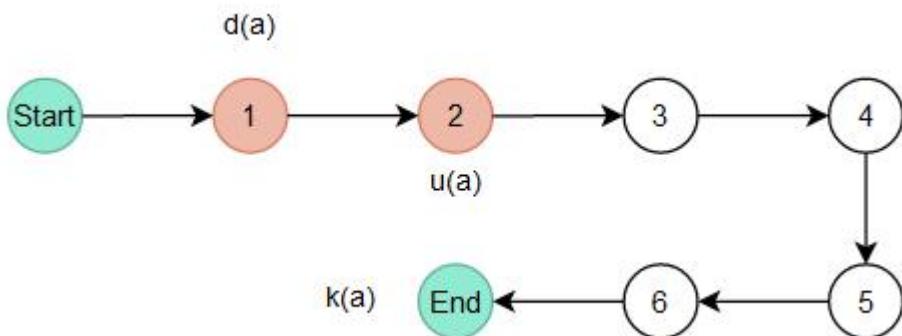
Table 6.29: Code Backend Xóa đơn hàng



Hình 6.29: Đồ thị dòng dữ liệu Xóa đơn hàng

Kiểm thử dòng sóng 7 biến: a(req), b(res), c(next), d(order), e(Order), f(id), g(success)

- **Kiểm thử dòng sóng biến a(req):**

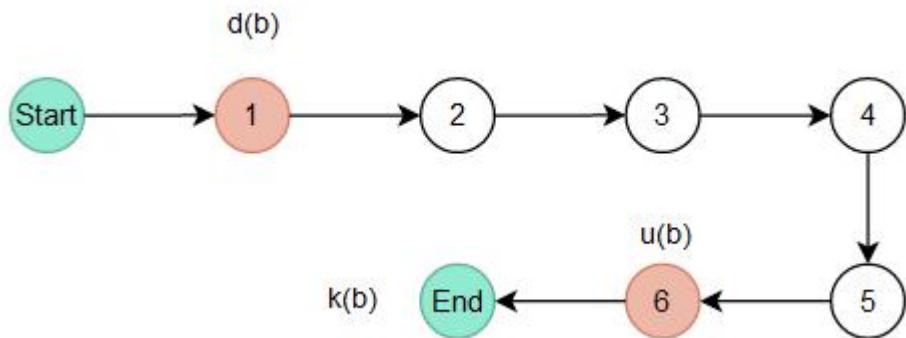


Hình 6.29.a: Đồ thị dòng sóng biến a(req) Xóa đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đòn bẩy biển $b(res)$:

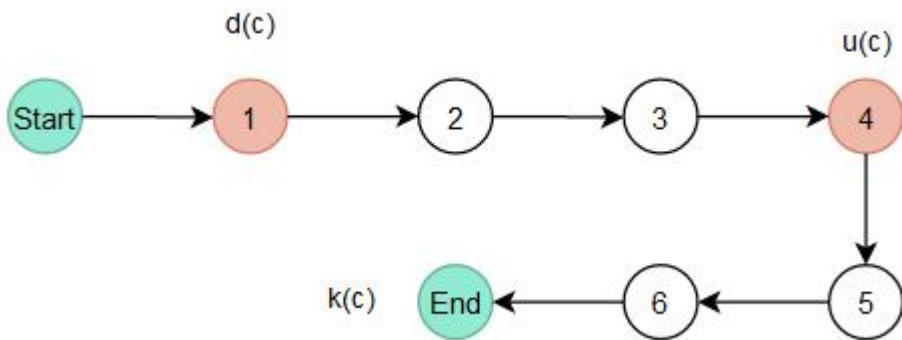


Hình 6.29.b: Đồ thị đòn bẩy biển $b(res)$ Xóa đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đòn bẩy biển $c(next)$:

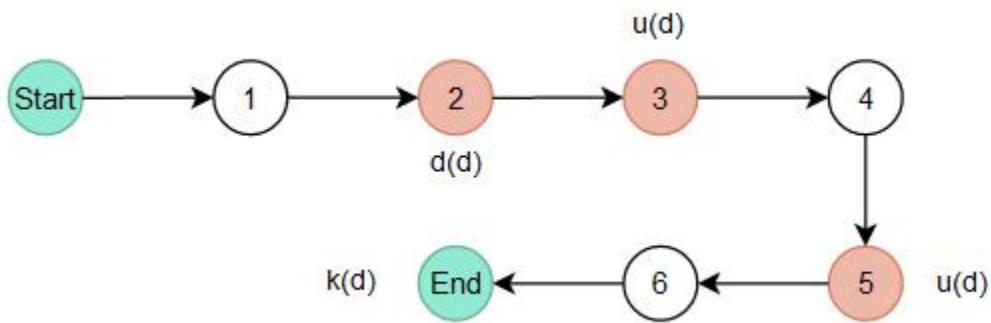


Hình 6.29.c: Đồ thị đòn bẩy biển $c(next)$ Xóa đơn hàng

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biên d(order):

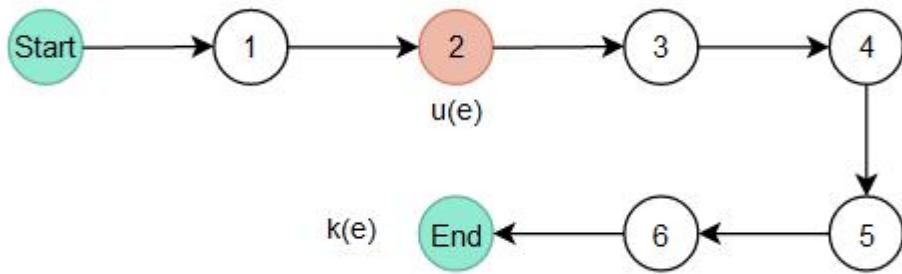


Hình 6.29.d: Đồ thị đời sóng biên d(order) Xóa đơn hàng

Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- Kiểm thử đời sóng biên e(Order):

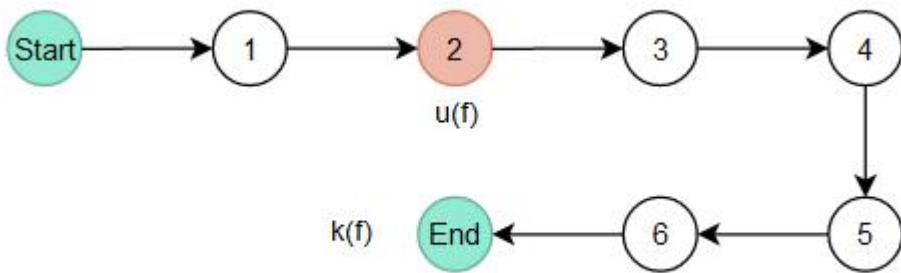


Hình 6.29.e: Đồ thị đời sóng biên e(Order) Xóa đơn hàng

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc ~u miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biên $f(id)$:

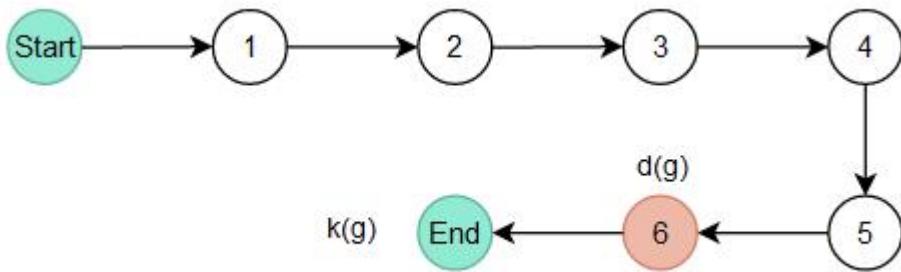


Hình 6.29.f: Đồ thị đời sóng biên $f(id)$ Xóa đơn hàng

Kịch bản 1: $\sim uk$

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biên $g(success)$:



Hình 6.29.g: Đồ thị đời sóng biên $g(success)$ Xóa đơn hàng

Kịch bản 1: $\sim dk$

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

30. Admin - See All Reviews of a Product

```

//Get All Reviews of A Product
exports.getProductReviews = catchAsyncError(async (req, res, next/*(1)*/) => {
  const product = await Product.findById(req.query.id); /*(2)*/
  if (!product /*(3)*/) {
  
```

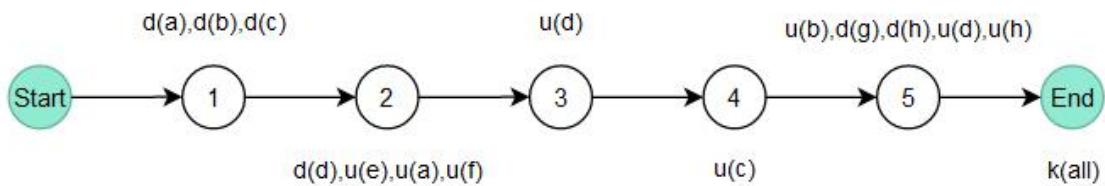
```

        return next(new ErrorHandler("Product not found", 404)); /*(4)*/
    }

    res.status(200).json({
        success: true,
        reviews: product.reviews,
   }); /*(5)*/
});

```

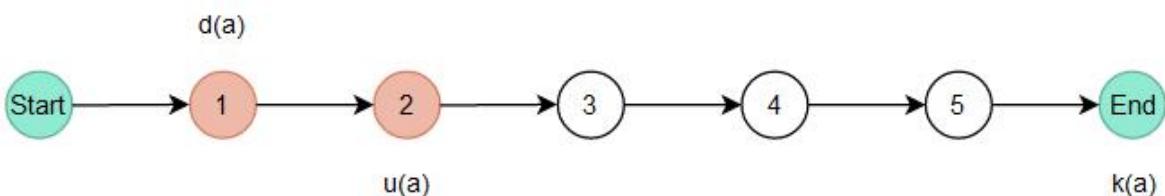
Table 6.30: Code Backend Xem tất cả các đánh giá của 1 sản phẩm



Hình 6.30: Đồ thị dòng dữ liệu Xem tất cả các đánh giá của 1 sản phẩm

Kiểm thử dòng 8 biến: a(req), b(res), c(next), d(product), e(Product), f(id), g(success), h(reviews)

- **Kiểm thử dòng 8 biến a(req):**

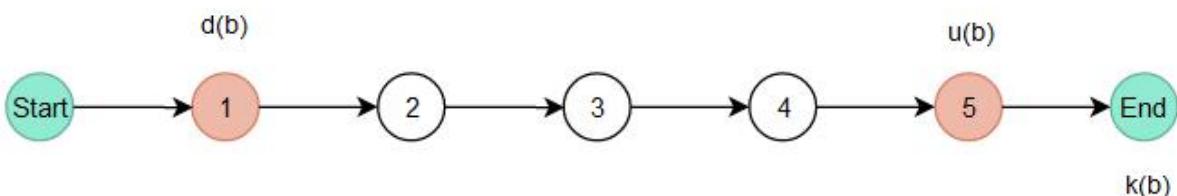


Hình 6.30.a: Đồ thị dòng 8 biến a(req) Xem tất cả các đánh giá của 1 sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử dòng 8 biến b(res):**

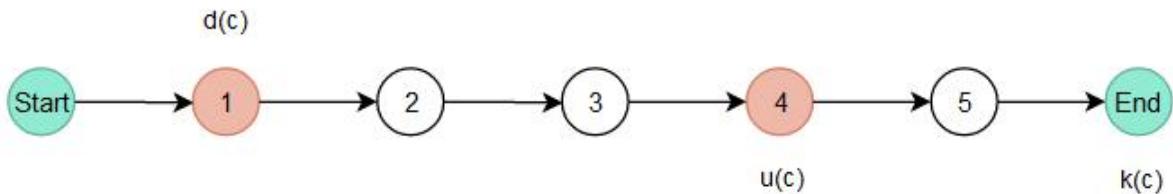


Hình 6.30.b: Đồ thị dòng sóng biến $b(res)$ Xem tất cả các đánh giá của 1 sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử dòng sóng biến $c(next)$:**

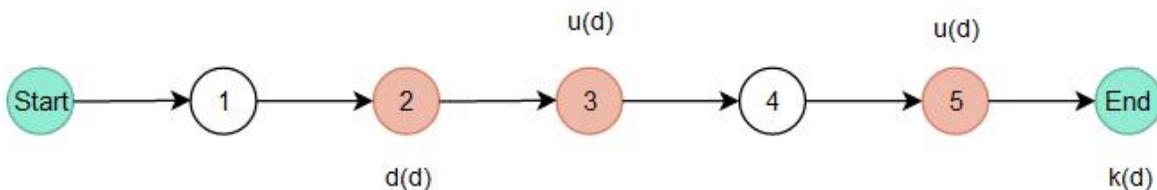


Hình 6.30.c: Đồ thị dòng sóng biến $c(next)$ Xem tất cả các đánh giá của 1 sản phẩm

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử dòng sóng biến $d(product)$:**

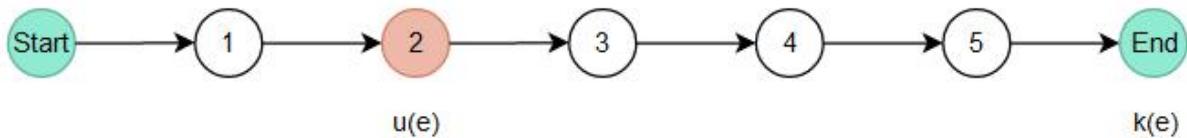


Hình 6.30.d: Đồ thị dòng sóng biến $d(product)$ Xem tất cả các đánh giá của 1 sản phẩm

Kịch bản 1: ~duuk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

- **Kiểm thử dòng sóng biến $e(Product)$:**

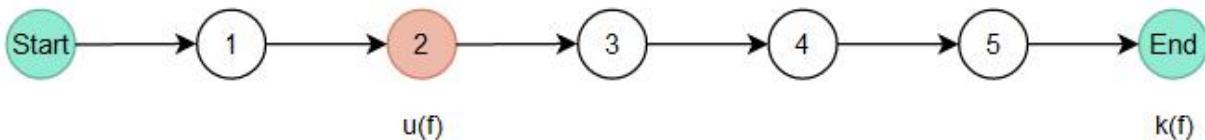


Hình 6.30.e: Đồ thị dòng sóng biến $e(Product)$ Xem tất cả các đánh giá của 1 sản phẩm

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- **Kiểm thử đời sống biển f(id):**

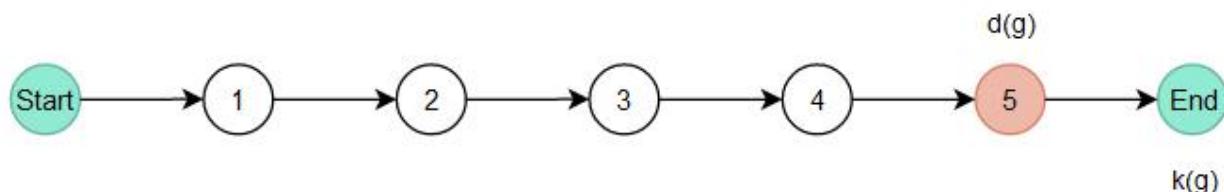


Hình 6.30.f: Đồ thị đời sống biển $f(id)$ Xem tất cả các đánh giá của 1 sản phẩm

Kịch bản 1: ~uk

=> **Kết luận:** Ở kịch bản 1, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- **Kiểm thử đời sống biển g(success):**

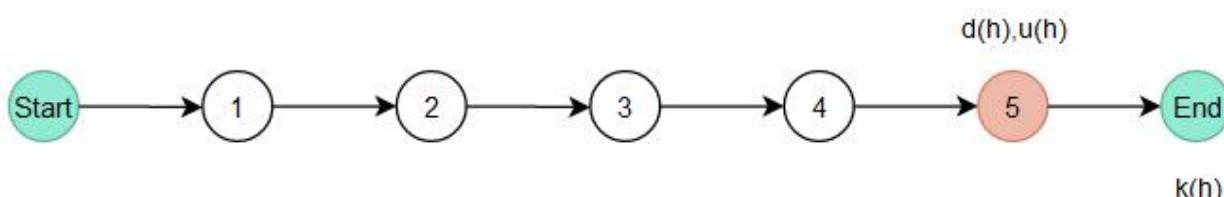


Hình 6.30.g: Đồ thị đời sống biển $g(success)$ Xem tất cả các đánh giá của 1 sản phẩm

Kịch bản 1: ~dk

=> **Kết luận:** Ở kịch bản 1, cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- **Kiểm thử đời sống biển h(reviews):**



Hình 6.30.h: *Đồ thị đồ thị sóng biển h(reviews) Xem tất cả các đánh giá của 1 sản phẩm*

Kịch bản 1: ~duk

=> **Kết luận:** Không có cặp đôi nào của Kịch bản 1 hoạt động bất thường.

31. Admin - Delete A Product Review

```
//Delete A Review
exports.deleteReview = catchAsyncError(async (req, res, next) => {
  const product = await Product.findById(req.query.productId); /*(2)*/
  if (!product /*(3)*/) {
    return next(new ErrorHandler("Product not found", 404)); /*(4)*/
  }

  const reviews = product.reviews.filter(
    (rev) => rev._id.toString() !== req.query.id.toString()
  ); /*(5)*/

  let avg = 0; /*(6)*/

  reviews.forEach((rev) => {
    avg += rev.rating;
  }); /*(7)*/

  let ratings = 0; /*(8)*/

  if (reviews.length === 0 /*(9)*/) {
    ratings = 0; /*(10)*/
  } else {
    ratings = avg / reviews.length; /*(11)*/
  }

  const numOfReviews = reviews.length; /*(12)*/

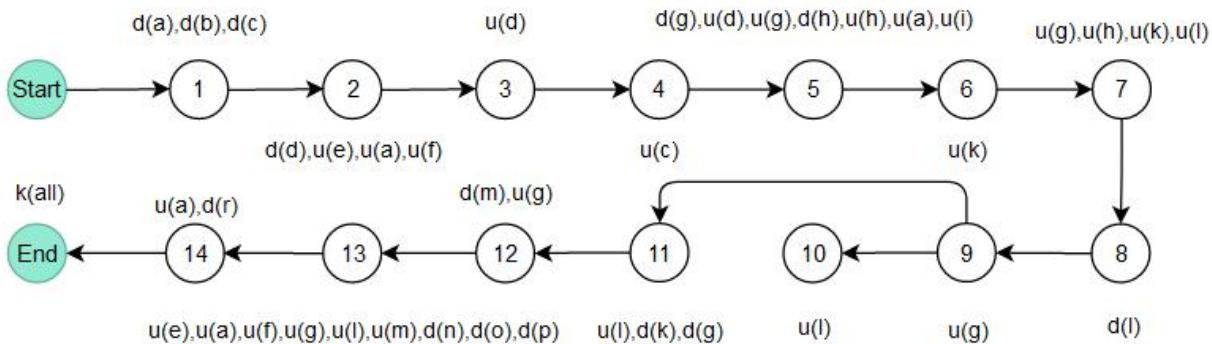
  await Product.findByIdAndUpdate(
    req.query.productId,
    {
      reviews,
      ratings,
      numOfReviews,
    },
    {
      new: true,
      runValidators: true,
      useFindAndModify: false,
    }
  );
});
```

```
    }
}); /*(13)*/

```

```
res.status(200).json({
  success: true,
}); /*(14)*/
```

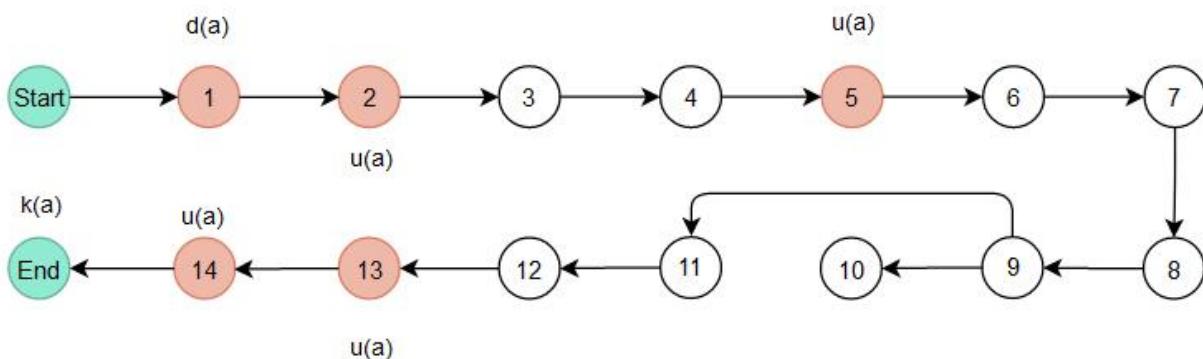
Table 6.31: Code Backend Xóa đánh giá



Hình 6.31: Đồ thị dòng dữ liệu Xóa đánh giá

Kiểm thử dòng sống 16 biến: *a(req)*, *b(res)*, *c(next)*, *d(product)*, *e(Product)*, *f(productId)*, *g(reviews)*, *h(rev)*, *i(id)*, *k(avgs)*, *l(ratings)*, *m(numOfReviews)*, *n(new)*, *o(runValidators)*, *p(useFindAndModify)*, *r(success)*

- Kiểm thử dòng sống biến *a(req)*:



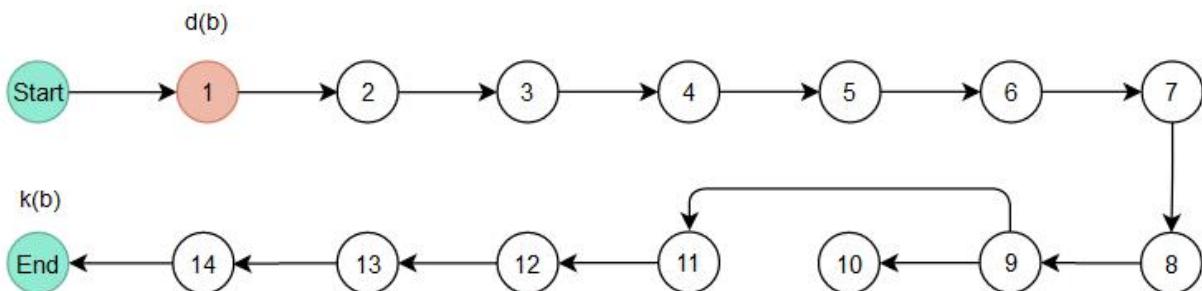
Hình 6.31.a: Đồ thị dòng sống biến *a(req)* Xóa đánh giá

Kịch bản 1: ~duuuuk

Kịch bản 2: ~duuuuk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- **Kiểm thử đời sóng biển b(res):**



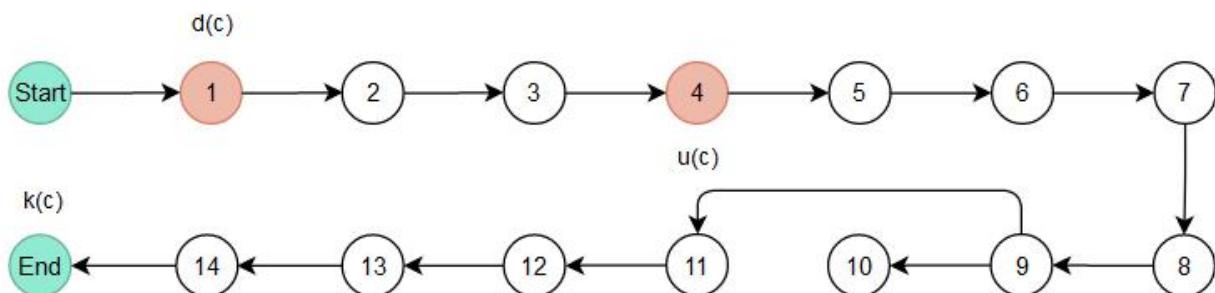
Hình 6.31.b: Đồ thị đời sóng biển b(res) Xóa đánh giá

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- **Kiểm thử đời sóng biển c(next):**



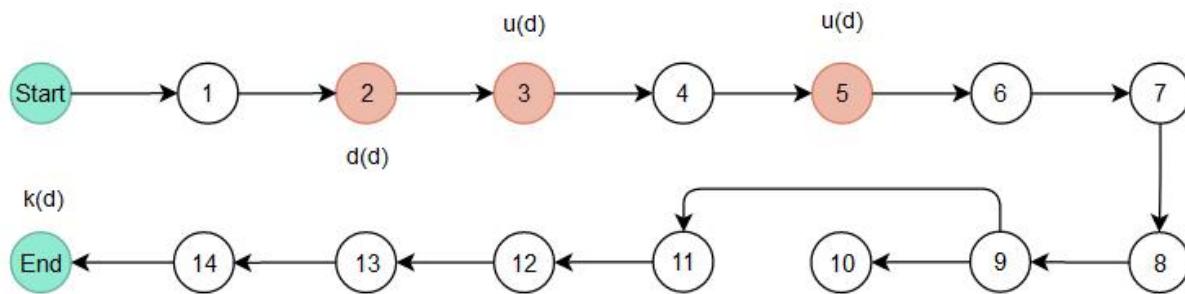
Hình 6.31.c: Đồ thị đời sóng biển c(next) Xóa đánh giá

Kịch bản 1: ~duk

Kịch bản 2: ~duk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sống biến d (product):



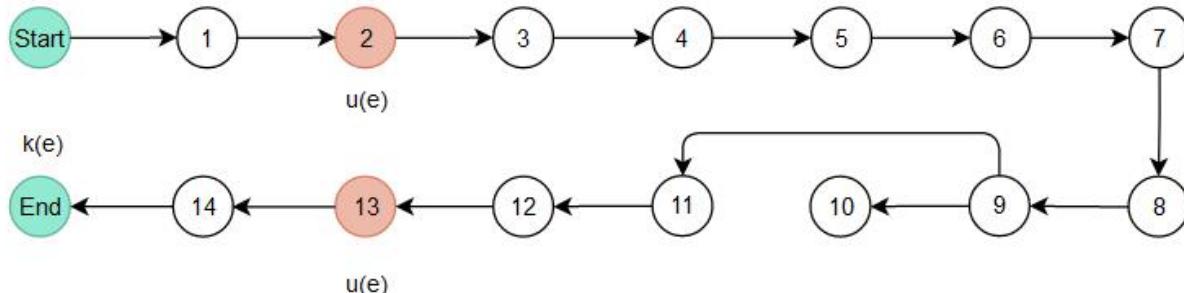
Hình 6.31.d: Đồ thị đời sống biến d (product) Xóa đánh giá

Kịch bản 1: ~duuk

Kịch bản 2: ~duuk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sống biến e (Product):



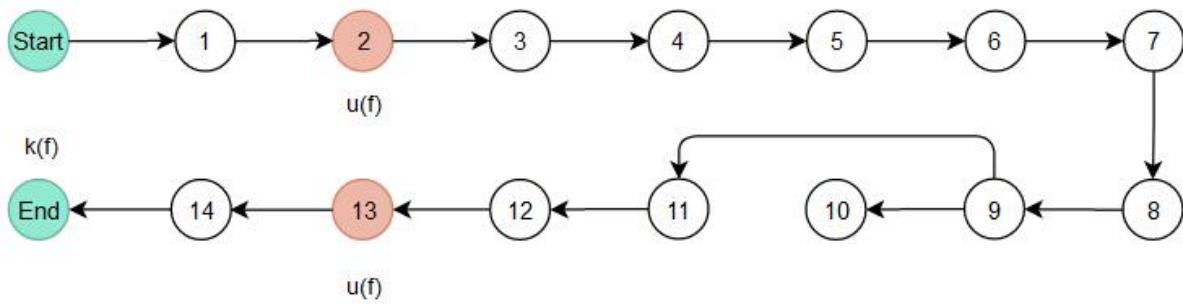
Hình 6.31.e: Đồ thị đời sống biến e (Product) Xóa đánh giá

Kịch bản 1: ~uuuk

Kịch bản 2: ~uuuk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biển $f(productId)$:



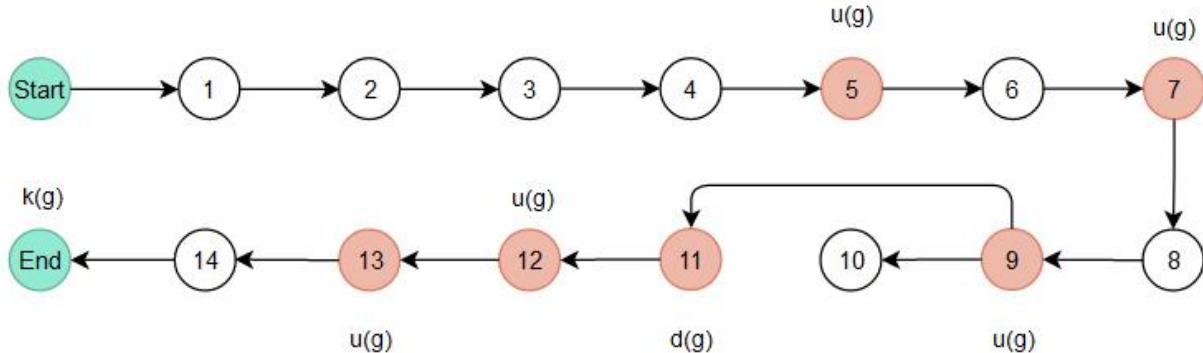
Hình 6.31.f: Đồ thị đời sóng biển $f(productId)$ Xóa đánh giá

Kịch bản 1: ~uuuk

Kịch bản 2: ~uuuk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biển $g(reviews)$:



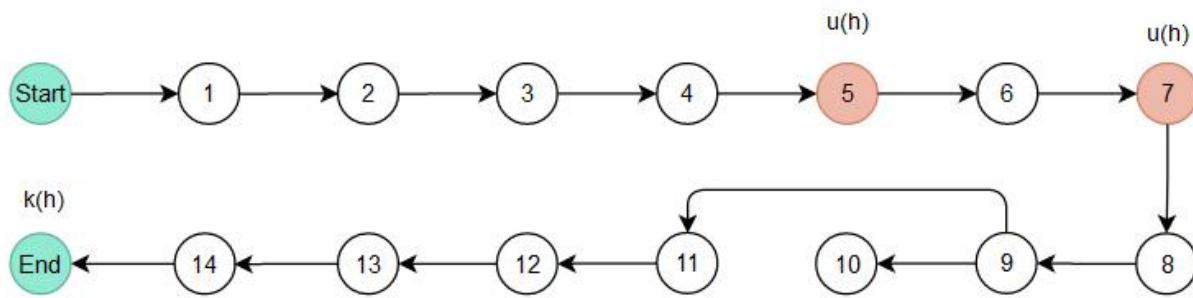
Hình 6.31.g: Đồ thị đời sóng biển $g(reviews)$ Xóa đánh giá

Kịch bản 1: ~uuuduuk

Kịch bản 2: ~uuuduuk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biển $h(\text{rev})$:



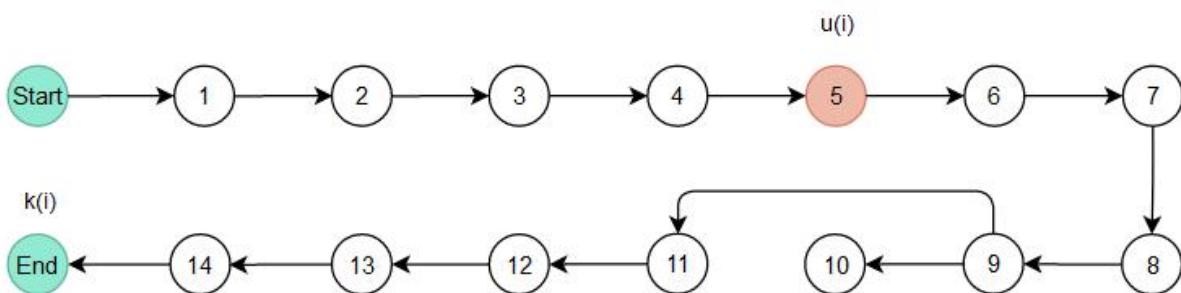
Hình 6.31.h: Đồ thị đời sống biển $h(\text{rev})$ Xóa đánh giá

Kịch bản 1: ~uuk

Kịch bản 2: ~uuk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biển chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biển $i(id)$:



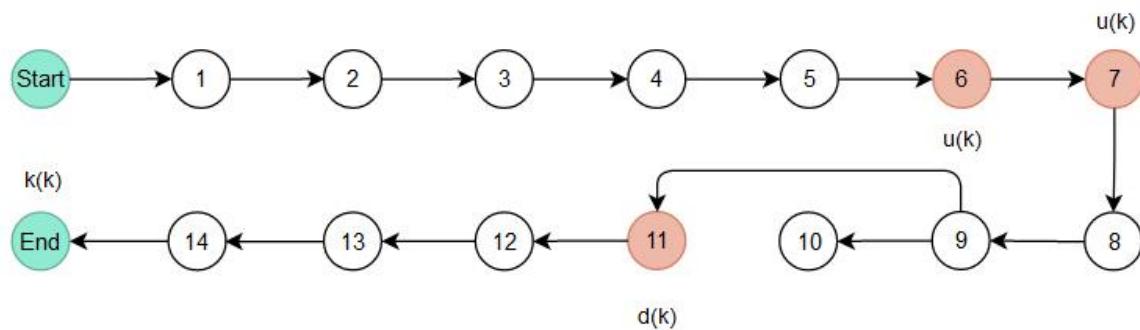
Hình 6.31.i: Đồ thị đời sống biển $i(id)$ Xóa đánh giá

Kịch bản 1: ~uuk

Kịch bản 2: ~uuk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biển chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biển $k(\text{avg})$:



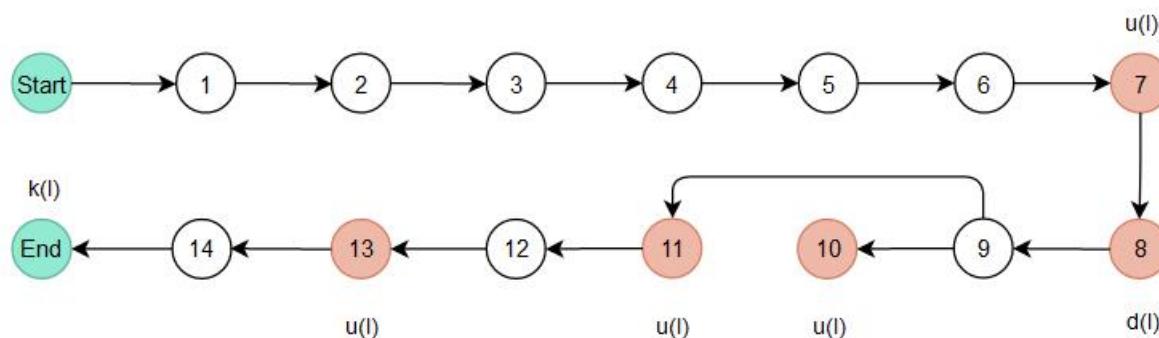
Hình 6.31.k: Đồ thị đời sóng biển $k(\text{avg})$ Xóa đánh giá

Kịch bản 1: ~uuuk

Kịch bản 2: ~uudk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào); và ở kịch bản 2 xuất hiện cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình; còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sóng biển $l(\text{ratings})$:



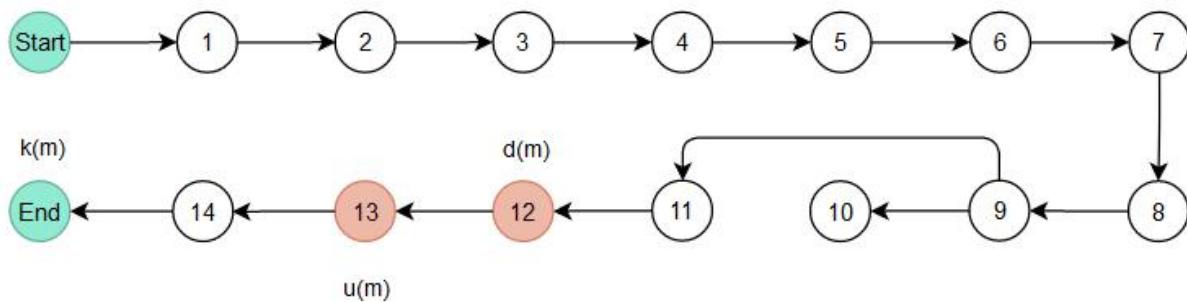
Hình 6.31.l: Đồ thị đời sóng biển $l(\text{ratings})$ Xóa đánh giá

Kịch bản 1: ~uduuk

Kịch bản 2: ~uduuk

=> **Kết luận:** Ở cả 2 kịch bản, việc $\sim u$ miêu tả trạng thái mà ở đó biến chưa tồn tại rồi được dùng ngay (trị nào), còn lại không có các cặp đôi khác hoạt động bất thường.

- Kiểm thử đời sống biển m (*numOfReviews*):



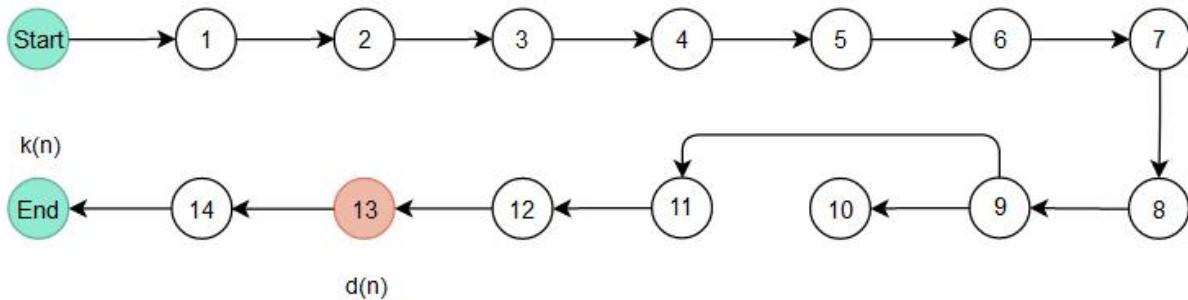
Hình 6.31.m: Đồ thị đời sống biển m (*numOfReviews*) Xóa đánh giá

Kịch bản 1: ~duk

Kịch bản 2: ~duk

=> **Kết luận:** Cả 2 kịch bản đều không có cặp đôi nào hoạt động bất thường.

- Kiểm thử đời sống biển n (*new*):



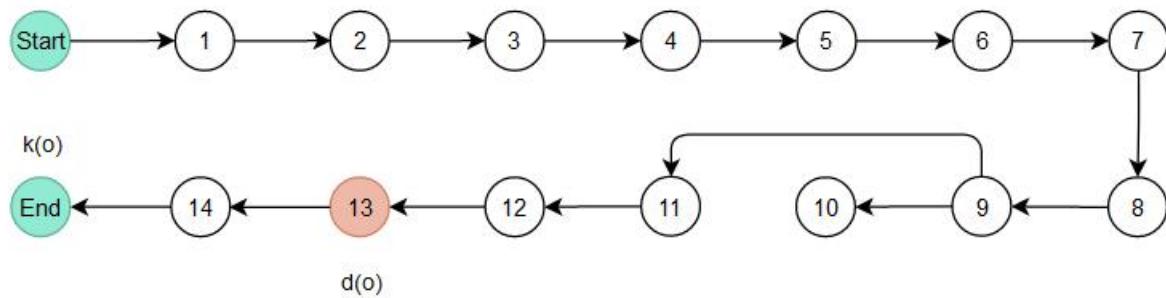
Hình 6.31.n: Đồ thị đời sống biển n (*new*) Xóa đánh giá

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi ***dk*** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến o (runValidators):



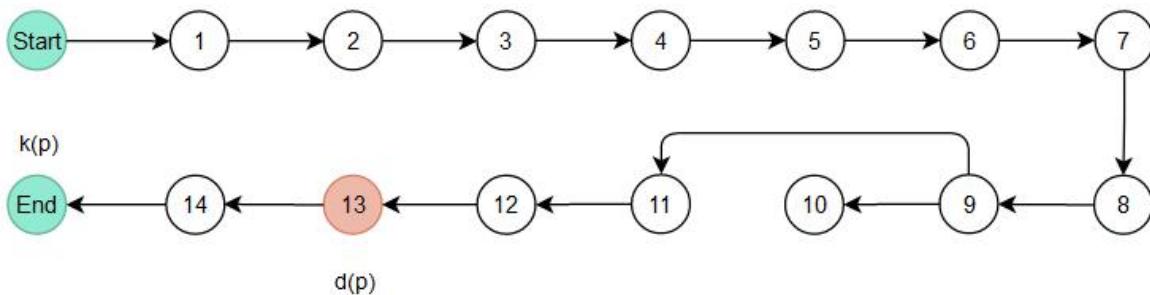
Hình 6.31.o: Đồ thị đời sống biến o (runValidators) Xóa đánh giá

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến p (useFindAndModify):



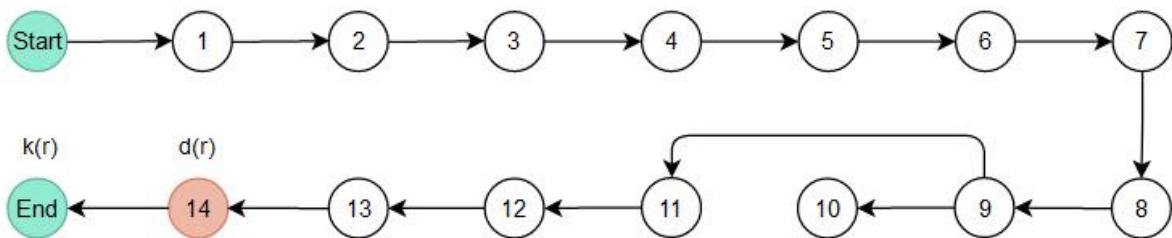
Hình 6.31.p: Đồ thị đời sống biến p (useFindAndModify) Xóa đánh giá

Kịch bản 1: ~dk

Kịch bản 2: ~dk

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi **dk** biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

- Kiểm thử đời sống biến r (success):



Hình 6.31.r: Đồ thị đời sống biến r (success) Xóa đánh giá

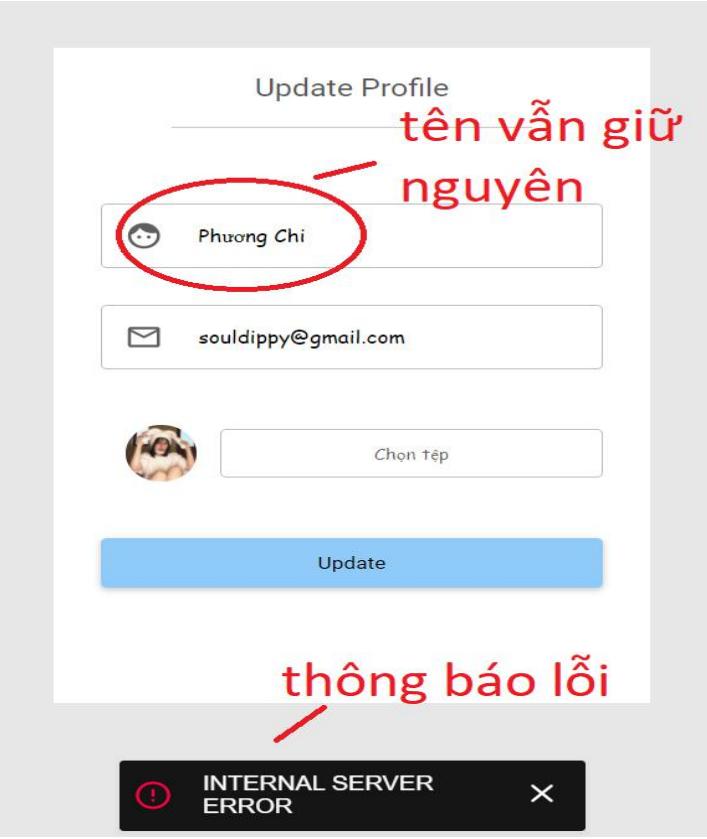
Kịch bản 1: $\sim dk$

Kịch bản 2: $\sim dk$

=> **Kết luận:** Ở cả 2 kịch bản đều xuất hiện cặp đôi dk biến được hình thành rồi xóa bỏ, hơi lạ, có thể đúng và chấp nhận được, nhưng có thể có lỗi lập trình.

CHƯƠNG VII: BUGS REPORT

1. Report 01

ID number	#01
Name	Lỗi cập nhật thông tin cá nhân
Reporter	Hoàng Minh Quang
Submit Date	18/11/2021
Summary	Lỗi không cập nhật được tên của user ở trang “Update Profile”
URL	https://new-bookstore-fieldproject.herokuapp.com/me/update
Screenshot	 A screenshot of a web application's 'Update Profile' page. At the top, it says 'Update Profile'. Below that is a form field containing the name 'Phương Chi', which is circled in red. To the right of the name is the text 'tên vẫn giữ nguyên' (name still remains). Below the name field is an email input field with the value 'souldippy@gmail.com'. Further down is a file upload section with a placeholder 'Chọn tệp' (Select file) and a profile picture thumbnail. At the bottom is a large blue 'Update' button. Red handwritten annotations are present: 'tên vẫn giữ nguyên' is written above the name field, and 'thông báo lỗi' (error message) is written above the 'INTERNAL SERVER ERROR' notification at the bottom.
Platform	Reactjs
Operating System	Window 10

Browser	Google Chrome
Severity	Major
Assigned to	Hoàng Minh Quang
Priority	Low

Table 7.1: Bug Report 01

Description

Khi vào trang “Update Profile” của user để cập nhật thông tin, tiến hành sửa thông tin của trường “Name” sau đó click “Update” thì thông báo lỗi và không cập nhật thông tin, thông tin trước đây vẫn giữ nguyên.

Steps to reproduce

- > Đăng nhập vào trang web bằng tài khoản user
- > Di chuyển chuột vào avatar của user.
- > Click vào “Profile”.
- > Xuất hiện trang My Profile.
- > Nhấn chọn “Edit Profile” để tiến hành chỉnh sửa.
- > Xuất hiện trang “Update Profile”.
- > Nhập tên muốn chỉnh sửa vào ô “Name” (Ở trường hợp này là “Thuỷ Chi”).
- > Nhấn Update để cập nhật.

Expected result

Chỉnh sửa thành công trường “Name” của “User Profile”.

Actual result

Chỉnh sửa thất bại, hiển thị hộp thoại lỗi “INTERNAL SERVER ERROR”.

Notes

2. Report 02

ID number	#02
Name	Lỗi quên mật khẩu thất bại
Reporter	Hoàng Minh Quang
Submit Date	19/11/2021

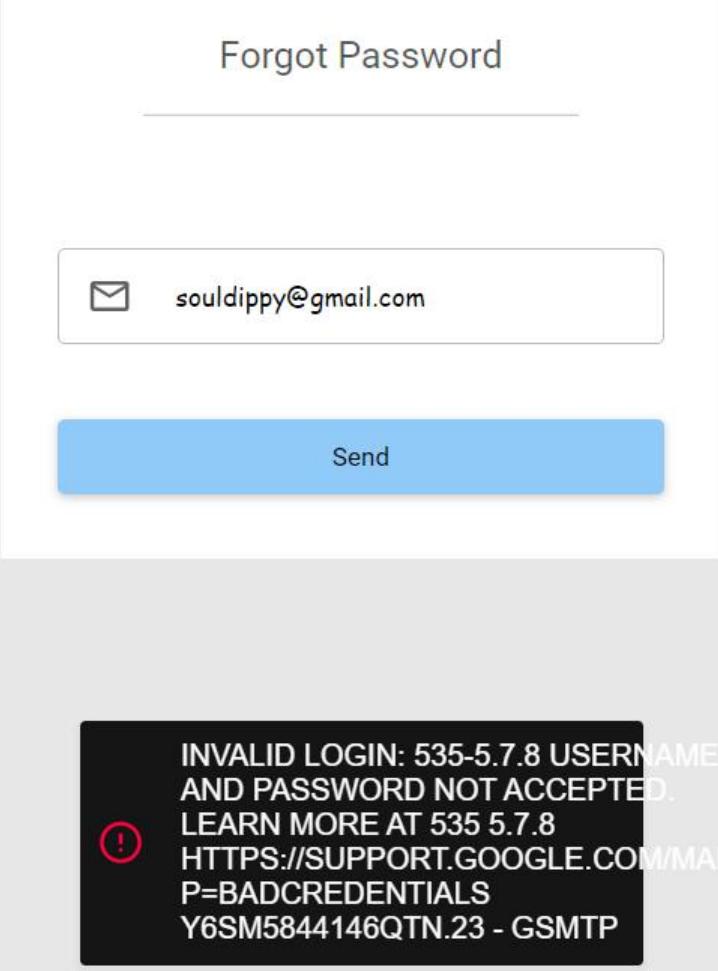
Summary	Lỗi xảy ra khi thực hiện yêu cầu cấp lại mật khẩu qua gmail ở trang “Forgot Password”
URL	https://new-bookstore-fieldproject.herokuapp.com/password/forgot
Screenshot	 <p>The screenshot shows a 'Forgot Password' form with an input field containing 'souldippy@gmail.com' and a blue 'Send' button below it. A black callout box with white text and a red exclamation mark icon appears, displaying the error message: 'INVALID LOGIN: 535-5.7.8 USERNAME AND PASSWORD NOT ACCEPTED. LEARN MORE AT 535 5.7.8 HTTPS://SUPPORT.GOOGLE.COM/MAIL/?P=BADCREDENTIALS Y6SM5844146QTN.23 - GSMTTP'.</p>
Platform	Reactjs
Operating System	Window 10
Browser	Microsoft edge
Severity	Major
Assigned to	Hoàng Minh Quang
Priority	Medium

Table 7.2: Bug Report 02

Description

Khi vào trang “Forgot Password” và nhập email đã đăng ký trước đó để tiến hành cấp lại mật khẩu email thì thông báo lỗi và quên mật khẩu thất bại.

Steps to reproduce

- > Truy cập vào trang login của trang web.
- > Click vào “Forgot Password”.
- > Xuất hiện trang “Forgot Password”.
- > Nhập email cần quên mật khẩu (ở trường hợp này là “souldippy@gmail.com”).
- > Click vào button “Send”.

Expected result

Email chứa link để đổi mật khẩu được gửi thành công.

Actual result

Gửi thất bại, hiển thị hộp thoại thông báo lỗi.

Notes

3. Report 03

ID number	#03
Name	Không thêm được sản phẩm mới vào database
Reporter	Hoàng Minh Quang
Submit Date	15/11/2021
Summary	Lỗi xảy ra khi thêm sản phẩm mới vào database
URL	https://new-bookstore-fieldproject.herokuapp.com/admin/product

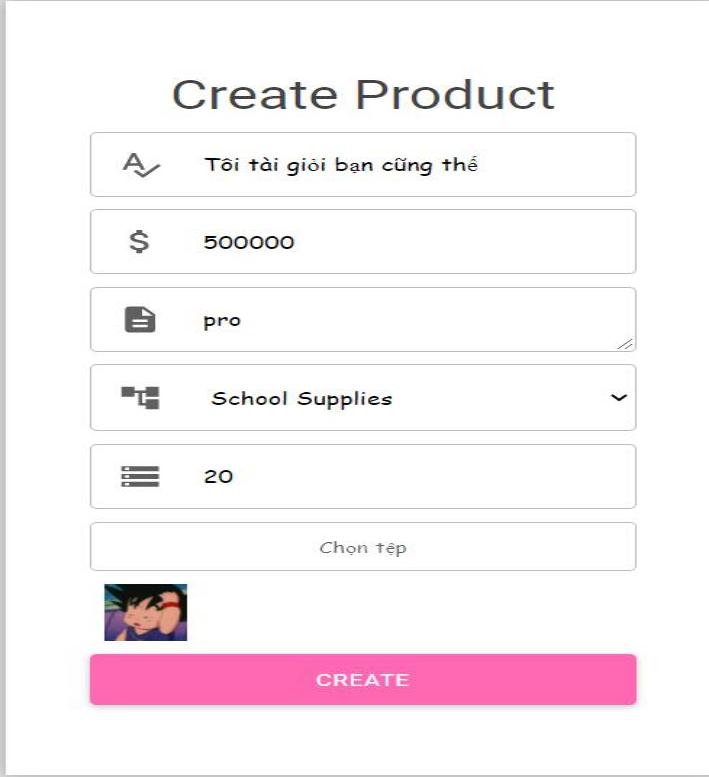
											
Screenshot	<p style="color: red; font-size: 1.5em;">action thông báo lỗi thêm sản phẩm thất bại</p> <p style="font-size: 0.8em;">Inspector</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">filter...</th> <th style="width: 80%; text-align: center;">Commit</th> </tr> </thead> <tbody> <tr> <td>NEW_PRODUCT_REQUEST</td> <td>+00:28.79</td> </tr> <tr> <td>NEW_PRODUCT_FAIL</td> <td>+00:01.52</td> </tr> <tr> <td>NEW_PRODUCT_REQUEST</td> <td>+00:56.93</td> </tr> <tr> <td style="outline: 2px solid red; border-radius: 50%; padding: 2px;">NEW_PRODUCT_FAIL</td> <td>+00:00.83</td> </tr> </tbody> </table>	filter...	Commit	NEW_PRODUCT_REQUEST	+00:28.79	NEW_PRODUCT_FAIL	+00:01.52	NEW_PRODUCT_REQUEST	+00:56.93	NEW_PRODUCT_FAIL	+00:00.83
filter...	Commit										
NEW_PRODUCT_REQUEST	+00:28.79										
NEW_PRODUCT_FAIL	+00:01.52										
NEW_PRODUCT_REQUEST	+00:56.93										
NEW_PRODUCT_FAIL	+00:00.83										
Platform	Reactjs										
Operating System	Window 10										
Browser	Microsoft edge										
Severity	Major										
Assigned to	Hoàng Minh Quang										
Priority	High										

Table 7.3: Bug Report 03

Description

Khi vào trang “Create Product” và tiến hành nhập các thông tin sản phẩm muốn tạo vào trong các trường bắt buộc và tiến hành tạo sản phẩm thì sản phẩm cần tạo không được thêm vào database. Trang redux hiển thị action “NEW_PRODUCT_FAIL” sau khi request.

Steps to reproduce

- > Đăng nhập vào trang web bằng tài khoản admin.
- > Di chuyển chuột vào avatar .
- > Click vào trang dashboard.
- > Nhấn vào “Product”.
- > Nhấn vào “Create” ở sidebar.
- > Hiển thị trang tạo sản phẩm “Create Product”.
- > Nhập các trường bắt buộc vào.
- > Click vào “Create”.

Expected result

Xuất hiện thông báo tạo sản phẩm thành công và sản phẩm được tạo sẽ lưu vào trong database.

Actual result

Tạo sản phẩm thất bại, không hiển thị thông báo lỗi.

Notes

4. Report 04

ID number	#05
Name	Không sửa được sản phẩm
Reporter	Nhi, Quang, Vũ
Submit Date	15/11/2021
Summary	Lỗi xảy ra khi sửa được sản phẩm
URL	https://new-bookstore-fieldproject.herokuapp.com/admin/product/id sản phẩm
Screenshot	

Update Product

A Agile Software Development: Princip

\$ 55471



Written by a software developer
for software developers. this



Science Technology Books

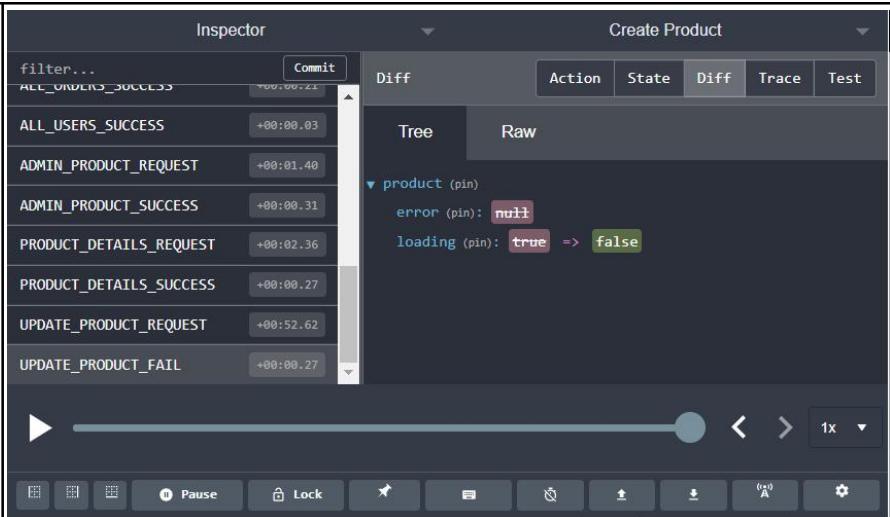


0

Choose Files



UPDATE



	Platform	Reactjs
	Operating System	Window 10
	Browser	Chrome
	Severity	Major
	Assigned to	Nhi, Quang, Vũ
	Priority	High

Table 7.4: Bug Report 04

Description

Khi vào trang “Update Product” và tiến hành xóa và nhập lại các thông tin sản phẩm muốn sửa vào trong các trường bắt buộc và tiến hành sửa sản phẩm thì sản phẩm cần sửa không được sửa trong database. Trang redux hiển thị action “UPDATE_PRODUCT_FAIL” sau khi request.

Steps to reproduce

- > Đăng nhập vào trang web bằng tài khoản admin.
- > Di chuyển chuột vào avatar .
- > Click vào trang dashboard.
- > Nhấn vào “Product”.
- > Nhấn vào icon “Modify” của 1 sản phẩm ở bảng tất cả các sản phẩm.
- > Hiển thị trang sửa sản phẩm “Update Product”.
- > Xóa và nhập lại các trường cần sửa vào.
- > Click vào “Update”.

Expected result

Xuất hiện thông báo chỉnh sửa sản phẩm thành công và sản phẩm được cập nhật sẽ lưu vào trong database.

Actual result

Sửa sản phẩm thất bại, không hiển thị thông báo lỗi.

Notes

5. Report 05

ID number	#05																																																							
Name	Không xóa được sản phẩm trong database																																																							
Reporter	Nhi, Quang, Vũ																																																							
Submit Date	15/11/2021																																																							
Summary	Lỗi xảy ra khi xóa sản phẩm trong database																																																							
URL	https://new-bookstore-fieldproject.herokuapp.com/admin/products																																																							
Screenshot	<p style="text-align: center;">ALL PRODUCTS</p> <table border="1"> <thead> <tr> <th>Product ID</th> <th>Name</th> <th>Stock</th> <th>Price</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>616943d5b4754e78a6189cab</td> <td>SQL QuickStart Guide: The Simplified Beginner's Guide to Managi...</td> <td>10</td> <td>15,462</td> <td> </td> </tr> <tr> <td>61694435b4754e78a6189cad</td> <td>Agile Software Development: Principles, Patterns, and Practices - ...</td> <td>0</td> <td>55,471</td> <td> </td> </tr> <tr> <td>616c42556208eba86f0ac235</td> <td>High Sierra Loop-Backpack</td> <td>10</td> <td>40,000</td> <td> </td> </tr> <tr> <td>616c428e6208eba86f0ac237</td> <td>Draw Your Own Comics</td> <td>10</td> <td>30,000</td> <td> </td> </tr> <tr> <td>6168669b56af38fcc86f9e6</td> <td>Artificial Intelligence Basics: A Non-Technical Introduction - Tom ...</td> <td>10</td> <td>51,588</td> <td> </td> </tr> <tr> <td>6169427eb4754e78a6189ca5</td> <td>Cloud Computing: Concepts, Technology, & Architecture - Thoma...</td> <td>19</td> <td>44,141</td> <td> </td> </tr> <tr> <td>61694320b4754e78a6189ca7</td> <td>The C Programming Language, 2nd Edition - Brian Kernighan</td> <td>18</td> <td>46,056</td> <td> </td> </tr> <tr> <td>616944aeb4754e78a6189cb0</td> <td>C++: The Complete Reference, 4th Editions - Herbert Schildt</td> <td>10</td> <td>84,608</td> <td> </td> </tr> <tr> <td>616c41e96208eba86f0ac233</td> <td>(ISC)2 CISSP Certified Information Systems Security Professional...</td> <td>20</td> <td>189,600</td> <td> </td> </tr> <tr> <td>616943db4754e78a6189ca9</td> <td>JavaScript: The Good Parts: The Good Parts - Douglas Crockford</td> <td>99</td> <td>16,300</td> <td> </td> </tr> </tbody> </table>	Product ID	Name	Stock	Price	Action	616943d5b4754e78a6189cab	SQL QuickStart Guide: The Simplified Beginner's Guide to Managi...	10	15,462		61694435b4754e78a6189cad	Agile Software Development: Principles, Patterns, and Practices - ...	0	55,471		616c42556208eba86f0ac235	High Sierra Loop-Backpack	10	40,000		616c428e6208eba86f0ac237	Draw Your Own Comics	10	30,000		6168669b56af38fcc86f9e6	Artificial Intelligence Basics: A Non-Technical Introduction - Tom ...	10	51,588		6169427eb4754e78a6189ca5	Cloud Computing: Concepts, Technology, & Architecture - Thoma...	19	44,141		61694320b4754e78a6189ca7	The C Programming Language, 2nd Edition - Brian Kernighan	18	46,056		616944aeb4754e78a6189cb0	C++: The Complete Reference, 4th Editions - Herbert Schildt	10	84,608		616c41e96208eba86f0ac233	(ISC)2 CISSP Certified Information Systems Security Professional...	20	189,600		616943db4754e78a6189ca9	JavaScript: The Good Parts: The Good Parts - Douglas Crockford	99	16,300	
Product ID	Name	Stock	Price	Action																																																				
616943d5b4754e78a6189cab	SQL QuickStart Guide: The Simplified Beginner's Guide to Managi...	10	15,462																																																					
61694435b4754e78a6189cad	Agile Software Development: Principles, Patterns, and Practices - ...	0	55,471																																																					
616c42556208eba86f0ac235	High Sierra Loop-Backpack	10	40,000																																																					
616c428e6208eba86f0ac237	Draw Your Own Comics	10	30,000																																																					
6168669b56af38fcc86f9e6	Artificial Intelligence Basics: A Non-Technical Introduction - Tom ...	10	51,588																																																					
6169427eb4754e78a6189ca5	Cloud Computing: Concepts, Technology, & Architecture - Thoma...	19	44,141																																																					
61694320b4754e78a6189ca7	The C Programming Language, 2nd Edition - Brian Kernighan	18	46,056																																																					
616944aeb4754e78a6189cb0	C++: The Complete Reference, 4th Editions - Herbert Schildt	10	84,608																																																					
616c41e96208eba86f0ac233	(ISC)2 CISSP Certified Information Systems Security Professional...	20	189,600																																																					
616943db4754e78a6189ca9	JavaScript: The Good Parts: The Good Parts - Douglas Crockford	99	16,300																																																					

Platform	Reactjs
Operating System	Window 10
Browser	Chrome
Severity	Major
Assigned to	Nhi, Quang, Vũ
Priority	High

Table 7.5: Bug Report 05

Description

Khi vào trang “Products” và chọn xóa sản phẩm thì sản phẩm cần xóa không được xóa khỏi database. Trang redux hiển thị action “DELETE_PRODUCT_FAIL” sau khi request.

Steps to reproduce

- > Đăng nhập vào trang web bằng tài khoản admin.
- > Di chuyển chuột vào avatar .
- > Click vào trang dashboard.
- > Nhấn vào “Product”.
- > Nhấn vào “All” ở sidebar.
- > Hiển thị trang tất cả các sản phẩm.
- > Click vào icon “Delete” của sản phẩm muốn xóa.

Expected result

Xuất hiện thông báo xóa sản phẩm thành công và sản phẩm được xóa khỏi database.

Actual result

Xóa sản phẩm thất bại, không hiển thị thông báo lỗi.

Notes

TÀI LIỆU THAM KHẢO

[1]. My Nguyen. (2020, March 19). Tìm hiểu về Unit Testing.

<https://viblo.asia/p/tim-hieu-ve-unit-testing-bWrZn7wrlxw>

[2]. My Nguyen. (2020, April 20). Tìm hiểu về Kiểm thử tích hợp.

<https://viblo.asia/p/tim-hieu-ve-kiem-thu-tich-hop-integration-testing-yMnKM94AK7P>

[3]. Phan Thi Duy Huyen. (2020, August 20). Kiểm thử hệ thống - System Testing

<https://viblo.asia/p/kiem-thu-he-thong-system-testing-Az45bD8oZxY>

[4]. Anh Tester. (2020, June 11). Kiểm thử hệ thống - Acceptance Testing - Kiểm thử chấp nhận

<https://anhtester.com/blog/acceptance-testing-kiem-thu-chap-nhan-b298.html>

----- HẾT -----