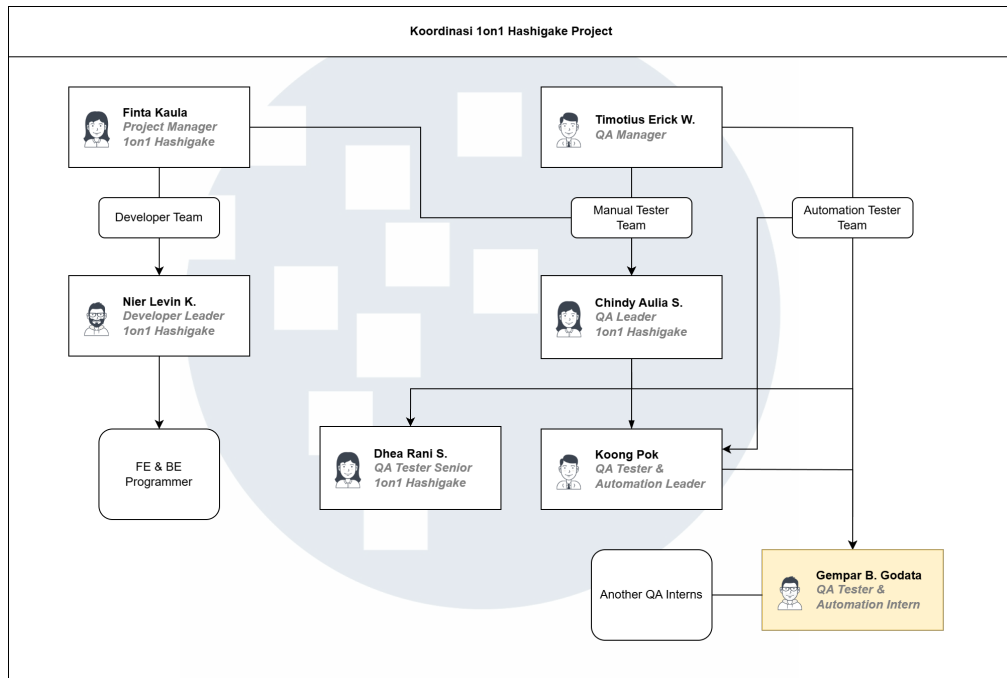


BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

3.1.1 Metode Koordinasi Proyek 1on1 Hashigake



Gambar 3.1 Diagram Koordinasi Kerja di Proyek 1on1 Hashigake

Proyek yang ditugaskan dalam kerja magang ini bernama resmi 1on1 Matching (Hashigake). Dalam proyek 1on1 Hashigake, QA Tester & Automation Intern memiliki peran sebagai bagian dari tim pengujian yang bertujuan untuk memastikan kualitas produk perangkat lunak sebelum dirilis. QA Tester & Automation Intern bekerja di bawah arahan langsung Timotius Erick selaku QA Manager menjadi supervisor, QA Leader (Chindy Aulia S.) dan didukung oleh QA Tester Senior (Dhea Rani S.), Automation Leader (Koong Pok) dan anggota divisi QA lain. Kedudukan ini berada dalam tim yang terpisah dari Developer Team, namun berkolaborasi erat untuk menangani pengujian manual maupun otomatisasi pada website Hashigake. Seluruh status pengerjaan tugas di dalam proyek ini dipimpin oleh Finta Kaula sebagai Project

Manager. Project Manager diberikan tanggung jawab untuk menjembatani tim proyek dengan Product Owner, Hirai-san.

Sebagai bagian dari tim pengujian, QA Tester & Automation Intern bertanggung jawab atas pelaksanaan skenario pengujian yang telah dirancang, termasuk penulisan dan eksekusi test case, pengelolaan bug, serta memastikan bahwa fitur yang diuji memenuhi standar kualitas yang ditetapkan client. Dalam proyek ini, mereka juga dilibatkan dalam pengembangan otomatisasi pengujian menggunakan teknologi terkini untuk meningkatkan efisiensi dan efektivitas proses pengujian. Selain itu, intern juga memiliki kesempatan untuk mempelajari sistem yang kompleks dan mendapatkan pembinaan langsung dari para pemimpin tim.

Kegiatan koordinasi pengerjaan tugas dilakukan melalui rapat harian seluruh tim proyek 1on Hashigake pada pagi hari (Daily Stand Up) dan ketika sore hari (End of Day) bila perlu dan memungkinkan. Rapat dilakukan secara online sesuai materi yang perlu didiskusikan dan pihak-pihak yang terlibat. Seluruh karyawan juga dapat berkoordinasi berkelompok melalui platform komunikasi bisnis Slack, Discord, Google Meet, Lark dan Whatsapp. Meeting mingguan untuk seluruh tim QA di FOS diadakan setiap hari Senin. Koordinasi bersama CEO Halim dan Suwa-san dilakukan setiap hari Jumat untuk pelatihan budaya kerja dan membahas kondisi terbaru dari semua proyek yang digarap.

Dalam proses pengembangan fitur Hashigake hingga deployment ke environment production di FOS, terdapat serangkaian tahapan yang dilakukan untuk memastikan kualitas produk sebelum digunakan oleh pengguna. Tahapan dimulai dengan pengumpulan persyaratan oleh Project Manager atau Product Manager dari klien. Persyaratan ini kemudian diteruskan kepada tim pengembang untuk direalisasikan. Setelah pengembangan selesai, tim QA akan melakukan pengujian awal sesuai dengan skenario yang telah dirancang dalam test case. Pengujian ini bertujuan untuk memastikan bahwa setiap fitur yang dikembangkan telah memenuhi standar dan spesifikasi yang ditetapkan.

Jika hasil pengujian awal memuaskan, tim QA dan Product Manager akan melanjutkan ke tahap System Integration Testing (SIT), bekerja sama dengan tim internal untuk memastikan kompatibilitas antar komponen sistem. Setelah SIT berhasil, proyek dilanjutkan ke tahap User Acceptance Testing (UAT), yang mana QA Leader mempresentasikan hasil pengembangan kepada pengguna untuk mendapatkan persetujuan. Jika pengguna menyatakan bahwa fitur tersebut telah memenuhi semua kebutuhan, maka proyek siap untuk dideploy ke production sesuai dengan jadwal yang telah disepakati.

Selama magang di FOS, intern memiliki hak untuk mengajukan pertanyaan dan berkomunikasi langsung dengan supervisor, anggota tim, maupun karyawan dari berbagai departemen yang relevan dengan tugas yang sedang dijalankan. Dalam pelaksanaan magang, kebutuhan komunikasi ini dapat mencakup, misalnya, permohonan izin tidak hadir untuk keperluan akademik, permintaan penjelasan terkait sistem data kepada Back-end Developer, atau diskusi mengenai pengembangan tugas otomatisasi pengujian dengan QA Leader. Intern juga didorong untuk berinteraksi dengan tim lainnya guna memperoleh informasi yang dibutuhkan untuk mendukung penyelesaian tugas yang efektif, termasuk kolaborasi antar QA dan developer jika diperlukan. Hal ini mencerminkan budaya kerja di FOS yang terbuka, mendukung pembelajaran, dan memfasilitasi kolaborasi aktif.

Sepanjang proses kegiatan proyek Hashigake, QA bertanggung jawab untuk menyusun dan memperbarui test case yang mencakup berbagai skenario pengujian. Jika ditemukan masalah atau bug pada fitur di suatu environment selama pengujian, QA akan mendokumentasikannya ke dalam Bug List dan melaporkan masalah tersebut kepada developer melalui platform dan Project Manager. Setelah perbaikan dilakukan oleh tim developer, tim QA akan melakukan pengujian ulang. Tiket pada Bug List akan dipindahkan dari status "Deployed" ke "Done Testing" ketika semua pengujian selesai dan fitur dinyatakan siap untuk dinaikkan ke berikutnya.

3.2 Tugas dan Uraian Kerja Magang

Selama menjalani program magang di PT Fata Organa Solusi (FOS), berbagai tugas dan tanggung jawab diberikan sesuai dengan peran sebagai QA Tester & Automation Intern. Kegiatan ini mencakup proses onboarding, pengembangan otomatisasi pengujian, pelaksanaan pengujian manual dan regression test, hingga finalisasi proyek dan dokumentasi bug. Seluruh aktivitas magang dirancang dalam dokumen Plan dan Technical Specification Document untuk mendukung pemahaman teknis, meningkatkan efisiensi proses pengujian, serta berkontribusi langsung dalam penyelesaian proyek. Namun pemberian waktu penugasan dipengaruhi oleh kondisi klien yang sangat dinamis, sehingga pekerjaan yang sedang dikerjakan dapat ditunda untuk menyelesaikan target yang lebih diprioritaskan. Berikut adalah rangkuman rentang waktu dan tugas-tugas utama yang dilaksanakan selama periode magang:

Tabel 3.1 Realisasi Periode Kegiatan Penugasan Magang

No.	Mulai	Selesai	Kegiatan
Onboarding dan Pengenalan Perusahaan FOS			
1	01/07/2024	07/07/2024	Pengenalan lingkungan kerja, knowledge sharing, self-learning, dan dokumentasi pengujian awal.
Pengembangan Kode Otomasi Pengujian			
2	• 09/07/2024 • 01/10/2024	• 31/08/2024 • 16/12/2024	Pembuatan, debugging, dan pengujian skenario otomatisasi untuk fitur Hashigake menggunakan Selenium dan StateSXT.
Uji Fitur Baru dan Regression Test			
3	• 01/09/2024 • 10/12/2024	• 30/09/2024 • 27/12/2024	Pengujian manual regression test, pelaporan bug, dan dokumentasi hasil pengujian untuk fitur baru.

Program magang di PT Fata Organa Solusi melibatkan tiga tahap utama yang dilaksanakan secara terstruktur. Tahap pertama adalah Onboarding dan Pengenalan Perusahaan, yang berfokus pada pemahaman budaya kerja, sistem internal perusahaan, serta pengenalan proses pengujian perangkat lunak melalui

kegiatan *knowledge sharing* dan *self-learning*. Tahap ini memberikan dasar pengetahuan yang diperlukan sebelum masuk ke tugas-tugas teknis.

Tahap kedua adalah Pengembangan Kode Otomasi Pengujian, yang dilakukan dalam dua periode kerja. Pada tahap ini, peserta bertanggung jawab untuk membuat, *debugging*, serta menguji skenario otomatisasi untuk aplikasi Hashigake menggunakan framework Selenium dan StateSXT. Aktivitas ini bertujuan untuk meningkatkan efisiensi pengujian perangkat lunak dengan mengotomatisasi berbagai proses pengujian berulang yang sebelumnya dilakukan secara manual.

Tahap terakhir adalah Uji Fitur Baru dan Regression Test, di mana dilakukan pengujian manual untuk memastikan stabilitas dan kompatibilitas fitur baru dalam aplikasi Hashigake. Kegiatan ini mencakup pelaporan bug serta dokumentasi hasil pengujian, yang kemudian digunakan untuk mendukung pengembangan lebih lanjut aplikasi.

3.2.1 Onboarding dan Pengenalan Perusahaan FOS

Tahap onboarding di FOS memberikan pemahaman mendalam tentang struktur environment yang digunakan dalam pengembangan aplikasi Hashigake. Aplikasi ini memiliki beberapa environment yang memisahkan proses pengembangan, pengujian, dan peluncuran, yaitu Development Environment (Dev) untuk pengembangan fitur baru, Test Environment untuk pengujian lebih lanjut oleh tim QA, dan Staging Environment sebagai simulasi yang hampir identik dengan production. Pada awal Desember 2024, diperkenalkan Custom Dev Environment yang mendukung metode login baru menggunakan akun email kustom. Fitur ini menggantikan login berbasis SSO Microsoft atau Google, memberikan fleksibilitas yang lebih besar bagi pengguna yang biasanya menggunakan email kantor.

1on1 Matching Testcase - Release 2025/01/21

File Edit View Insert Format Data Tools Extensions Help

Menu

100%

123

Default

10

B

Z

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10

10</

Gambar 3.2 Lembar Table of Contents dari Dokumen Testcase 1on1 Matching

Dalam rangka mendukung kegiatan pengujian aplikasi Hashigake, tim QA FOS menggunakan dokumen test case yang dikelola secara internal menggunakan Google Sheet, sebagaimana terlihat pada Gambar 3.2. Dokumen ini dirancang dengan struktur yang rapi dan sistematis untuk memastikan semua skenario pengujian tercakup dengan baik. Setiap dokumen test case dinamai berdasarkan fungsinya, seperti untuk Release Hashigake pada tanggal 21 Januari 2025, serta untuk kebutuhan pengujian spesifik seperti Sanity Check dan Regression Testing (termasuk pengujian terkait migrasi server yang dilakukan pada bulan September).

Dokumen ini terdiri dari 41 lembar sheet, yang masing-masing mencakup skenario pengujian yang dibagi berdasarkan peran pengguna dan fitur aplikasi. Struktur penamaannya dirancang untuk mencerminkan fungsi dan cakupan pengujian, antara lain:

- 0.x: Mencakup testcase untuk fitur Login & Registrasi.
- 1.x: Mencakup skenario pengujian untuk fitur pengguna Normal User.
- 2.x: Mencakup pengujian untuk fitur Admin.
- 3.x: Mencakup pengujian untuk fitur Super Admin.
- 4.x: Digunakan untuk menguji skenario Negative Behavior Test.

Normal User pada Hashigake hanya memiliki akses terbatas ke tiga menu utama, yaitu Match List, User Profile, dan User Matching Report. Fitur-fitur ini dirancang untuk memenuhi kebutuhan dasar pengguna individu dalam menjelajahi dan memanfaatkan fungsi utama aplikasi. Admin memiliki akses yang lebih luas dibandingkan Normal User. Selain fitur-fitur yang dimiliki oleh Normal User, Admin juga memiliki akses ke menu tambahan, yaitu Admin Management (meliputi User List, Department, Holiday, dan Group Setting) dan Admin Report (khususnya Company-Wide User Matching Report) yang berfungsi untuk mengelola data dan laporan bagi perusahaan tempat mereka berada. Super Admin memiliki akses paling komprehensif di Hashigake. Selain menggabungkan semua menu yang tersedia untuk Normal User dan Admin, Super Admin memiliki akses eksklusif ke menu Matching Simulation. Peran ini memungkinkan Super Admin untuk melihat seluruh pengguna dari berbagai perusahaan yang berbeda, memberikan mereka kemampuan untuk mengelola data dan melakukan simulasi pencocokan secara menyeluruh di seluruh organisasi yang menggunakan layanan Hashigake.

No.	State 1	State 2	State 3	Details
1	1	1		scroll down matchlist page
2	1	2		view other user profile
3	1	3		view user own profile page
4	1	4		view user report page
5	1	5	3	view user own profile page by profile card
6	1	5	6	login
7	1	7	1	privacy policy page
8	1	8		Open FAQ

Elements	Appearance	Responsiveness	Hoverable	Clickable	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	Scenario
All Menu in Left Side Bar	☒	☒	☐	☒							
Profile button	☒	☒	☒	☒							
Profile Menu (Profile & Logout)	☒	☒	☒	☒							
See More button in user's own Profile card	☒	☒	☒	☒							
See More button in other user profile card	☒	☒	☒	☒							
User's Own Profile card	☒	☒	☐	☐							
Other user profile card	☒	☒	☐	☐							
1on1 Logo	☒	☒	☐	☐							

Gambar 3.3 Struktur Lembar Dokumen Testcase Fitur Match List

Struktur sebuah test case dalam proyek 1on1 Hashigake dirancang secara komprehensif untuk memastikan bahwa setiap fitur aplikasi berfungsi

dengan baik, baik secara fungsional maupun visual. Setiap test case mencakup beberapa komponen utama, yaitu Detail Test Case, yang berisi dokumentasi tentang transisi antar state aplikasi, seperti navigasi antar halaman atau aksi tertentu seperti login, melihat profil, atau mengakses laporan. Selain itu, terdapat deskripsi detail mengenai perilaku yang diharapkan dari sistem, seperti tampilan profil pengguna atau pengalihan ke halaman kebijakan privasi.

Setiap lembar sheet mencantumkan informasi detail mengenai jenis pengujian (Functional Testing dan System Testing), teknik pengujian yang digunakan (seperti Black Box Testing, Equivalence Partition, dan Boundary Analysis), serta rincian jumlah test case yang dijalankan. Contohnya, untuk fitur Register terdapat beberapa sub-fitur seperti "Privacy Policy Page", "Basic User Information Page", dan "Matching Frequency Page" yang dijalankan melalui berbagai kombinasi skenario input text dan file.

Yoni Matching Testcase - Release 2025/01/21

File Edit View Insert Format Data Tools Extensions Help

Menu

100%

Default

14

B

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

1332

1333

1334

1335

1336

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

1458

1459

1460

1461

1462

1463

1464

1465

1466

1467

1468

1469

1470

1471

1472

1473

1474

1475

1476

1477

1478

1479

1480

1481

1482

1483

1484

1485

1486

1487

1488

1489

1490

1491

1

Gambar 3.4 Contoh Skenario Pengujian Fitur Match List

Progres eksekusi test case seperti pada Gambar 3.4 dicatat dengan mencantumkan nama tester, tanggal pelaksanaan, dan hasil pengujian yang dikategorikan menjadi empat status utama: PASSED untuk hasil yang sesuai dengan ekspektasi, FAILED untuk hasil yang tidak sesuai, BUGFIX jika ditemukan bug yang perlu diperbaiki, dan HOLD untuk pengujian yang ditunda

karena ketergantungan atau fungsionalitas yang belum selesai. Untuk pengujian yang membutuhkan validasi eksternal, seperti integrasi data atau respons sistem eksternal, hasilnya dicatat secara terpisah di bawah External Check Result.

Setiap test case juga dilengkapi dengan tingkat prioritas, seperti High atau Low, yang menunjukkan tingkat kepentingan fungsi tersebut terhadap keseluruhan sistem. Diagram transisi state turut disertakan untuk menggambarkan alur navigasi dan logika fungsional yang diharapkan, sehingga konsistensi logika aplikasi dapat terjaga. Selain itu, dilakukan Appearance Check secara manual untuk memverifikasi elemen visual aplikasi, seperti responsivitas antarmuka, tampilan hover dan clickable pada tombol, serta keakuratan elemen seperti logo, kartu profil, dan menu.

Pendekatan terstruktur ini tidak hanya memastikan pengujian dilakukan secara menyeluruh, tetapi juga mendokumentasikan progres dan permasalahan secara jelas, sehingga memudahkan kolaborasi antara tim QA, developer, dan pemangku kepentingan lainnya dalam menjaga standar kualitas aplikasi yang tinggi sepanjang siklus pengembangan. QA Tester & Automation Intern memiliki akses ke berbagai API Endpoint berbasis Postman yang mendukung otomatisasi proses pengujian website Hashigake dan meningkatkan efisiensi kerja. API ini mencakup berbagai kebutuhan, mulai dari Survey API untuk pengelolaan survei pengguna, Setting API untuk pengaturan kebijakan, Report API untuk menghasilkan laporan pencocokan dan kepuasan pengguna, hingga Reminder Email API untuk mengirimkan pengingat harian. Selain itu, tersedia Recording API untuk perekaman rapat, Profile API untuk pengelolaan data profil pengguna, dan Matching Event API untuk mengelola acara pencocokan.

Kelompok API lainnya yang mendukung adalah Login API untuk autentikasi pengguna, Holiday API untuk pengelolaan data libur, serta Administration API yang memungkinkan pengelolaan data perusahaan, pengguna, grup, dan topik melalui sistem yang terintegrasi. API ini menjadi alat penting bagi tim QA untuk berinteraksi langsung dengan sistem aplikasi tanpa

perlu interaksi manual, mempercepat proses pengujian dan memastikan kualitas aplikasi tetap terjaga di setiap tahap pengembangan.

Melalui proses onboarding yang terstruktur, peserta magang tidak hanya memahami teknis penggunaan environment dan API, tetapi juga dilatih untuk bekerja secara kolaboratif dalam tim lintas fungsi untuk mencapai standar kualitas yang tinggi. Budaya kerja di PT Fata Organa Solusi (FOS) sangat menjunjung tinggi profesionalisme, kolaborasi, dan etika kerja yang sesuai dengan standar Jepang. Peserta magang diajarkan untuk mengutamakan disiplin waktu, akuntabilitas dalam setiap tugas, dan komitmen terhadap kualitas hasil kerja serta target waktu yang dijanjikan untuk menyelesaikan tugas.

Selama onboarding, peserta magang juga diperkenalkan pada nilai-nilai inti perusahaan, seperti inovasi, tanggung jawab, dan kerjasama tim. Budaya keterbukaan dalam komunikasi di FOS memungkinkan peserta magang untuk bebas bertanya, memberikan ide, atau berdiskusi dengan atasan dan anggota tim lainnya. Dengan suasana kerja yang inklusif dan suportif, setiap individu didorong untuk berkembang, baik dalam hal keterampilan teknis maupun kemampuan interpersonal.

Selain itu, peserta magang diperkenalkan pada sistem kerja yang terorganisir dengan baik, termasuk prosedur pelaporan, mekanisme penanganan masalah, dan penggunaan alat-alat komunikasi seperti Slack untuk koordinasi tim. Kombinasi dari pendekatan teknis yang mendalam dan budaya kerja yang kondusif membantu peserta magang mempersiapkan diri untuk menghadapi tantangan yang lebih kompleks selama menjalani tugas mereka di FOS.

3.2.2 Pengembangan Kode Otomasi Pengujian

Pengembangan kode otomasi pengujian dimulai dengan melakukan setup perangkat komputer yang digunakan untuk menjalankan framework StateSXT. Langkah ini mencakup instalasi perangkat lunak pendukung, pengaturan dependensi, dan konfigurasi agar framework dapat beroperasi dengan optimal. Setelah setup selesai, proses dilanjutkan dengan mempelajari

cara membuat kode otomatis yang mampu menjalankan skenario dari dokumen test case Hashigake. Hal ini penting untuk memastikan setiap fitur yang diuji dapat berfungsi dengan baik sesuai spesifikasi.

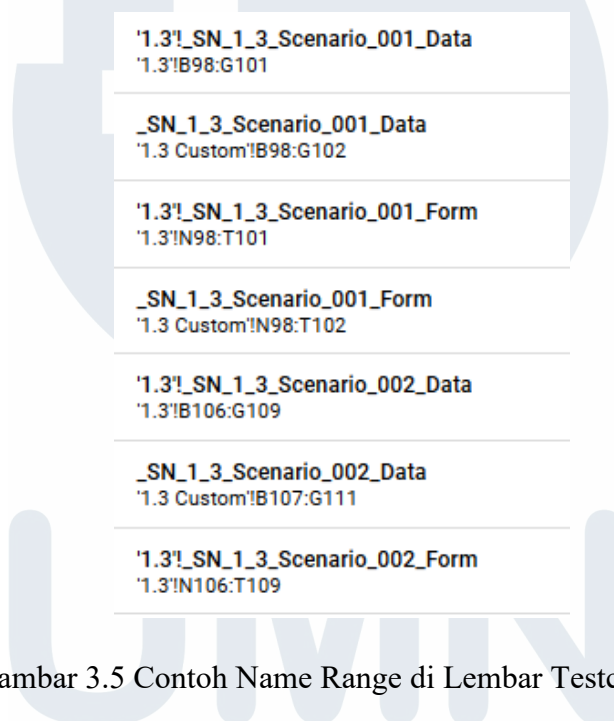
Dalam pengembangan otomatisasi pengujian, framework StateSXT, yang dikembangkan oleh Jason Caleb (karyawan FOS), menjadi solusi yang diterapkan di FOS. Framework ini dirancang untuk bekerja secara sinergis dengan Selenium Python, memungkinkan pengelolaan skenario pengujian yang lebih terstruktur dan mudah untuk dimodifikasi [20]. StateSXT mengimplementasikan State Design Pattern dengan cara memanfaatkan konsep state machine untuk memodelkan logika skenario pengujian, memungkinkan setiap langkah pengujian diuraikan dalam bentuk diagram yang mudah dipahami dan diimplementasikan.

Keunggulan StateSXT terletak pada kemampuannya menangani pengujian kompleks dengan jumlah skenario yang besar, seperti yang diterapkan pada aplikasi Hashigake. Dengan fitur-fitur yang disediakan, tim QA dapat dengan mudah memvisualisasikan, mengelola, dan memperbarui test case, bahkan ketika terjadi perubahan signifikan pada aplikasi. StateSXT juga menyediakan integrasi dengan Google Sheets dan Google Drive untuk manajemen data dan pelaporan yang lebih baik. Kombinasi antara Selenium Python dan StateSXT tidak hanya mempercepat proses pengujian, tetapi juga meningkatkan akurasi dalam mendeteksi bug dan meminimalkan risiko inkonsistensi dari pengujian oleh manusia.

Dokumen test automation memiliki perbedaan mendasar dengan dokumen pengujian manual. Pada automation test document, elemen-elemen yang diuji digambarkan dalam bentuk diagram, yang dapat diubah dan dimodifikasi untuk menyesuaikan logika otomatisasi sesuai kebutuhan. Diagram ini disebut State Transition Diagram (STD) dan menjadi inti dari logika pengujian otomatis. Sebuah STD dapat menggambarkan alur aksi sederhana antar elemen pada fitur web Hashigake atau bahkan alur kompleks yang menjelaskan relasi dan aksi antar elemen-elemen fitur yang lebih luas. Dengan

demikian, STD berfungsi sebagai panduan dalam menulis kode otomasi yang presisi dan efektif.

Setiap kode otomasi yang dibuat mengikuti logika dan skenario yang telah digambarkan dalam STD. Diagram ini menjadi acuan utama untuk memastikan bahwa setiap langkah yang dilakukan oleh kode otomasi telah sesuai dengan skenario pengujian yang dirancang sebelumnya. Hal ini memberikan jaminan bahwa setiap aksi yang dilakukan otomatisasi sesuai dengan alur logis aplikasi dan memenuhi standar kualitas yang ditetapkan.



'1.3'!_SN_1_3_Scenario_001_Data	'1.3'!B98:G101
_SN_1_3_Scenario_001_Data	'1.3 Custom'!B98:G102
'1.3'!_SN_1_3_Scenario_001_Form	'1.3'!N98:T101
_SN_1_3_Scenario_001_Form	'1.3 Custom'!N98:T102
'1.3'!_SN_1_3_Scenario_002_Data	'1.3'!B106:G109
_SN_1_3_Scenario_002_Data	'1.3 Custom'!B107:G111
'1.3'!_SN_1_3_Scenario_002_Form	'1.3'!N106:T109

Gambar 3.5 Contoh Name Range di Lembar Testcase 1.3

Lembar kerja test case yang diotomasi diberi nama sesuai dengan judul test case yang ada pada folder Page StateSXT. Penamaan ini mempermudah identifikasi dan pengelolaan dokumen test case yang telah diotomasi, terutama ketika test case tersebut perlu diperbarui atau dimodifikasi. Proses ini memastikan bahwa setiap test case yang diotomasi dapat dilacak dengan mudah dan tetap relevan seiring dengan perubahan yang mungkin terjadi pada fitur aplikasi Hashigake.

Persiapan penggunaan framework StateSXT dimulai dengan membuat folder baru sebagai direktori kerja, kemudian masuk ke dalamnya. Langkah

berikutnya adalah menjalankan perintah `pip install statesxt` untuk menginstal paket StateSXT dan `statesxt generate` untuk menghasilkan template proyek. Setelah itu, dibuat dan diaktifkan virtual environment menggunakan Poetry sebagai manajer dependensi Python. Instalasi Poetry dilakukan dengan perintah `pip install poetry`, dilanjutkan dengan konfigurasi menggunakan `poetry config virtualenvs.in-project true`. Selanjutnya, perintah `poetry install --no-root` digunakan untuk menginstal semua dependensi proyek.

```

corporatematching-selenium -
11 python = "^3.10"
12 allure-pytest = "2.13.2"
13 allure-python-commons = "2.13.2"
14 async-generator = "1.10"
15 attrs = "23.1.0"
16 bcrypt = "4.0.1"
17 blinker = "1.6.3"
18 brotli = "1.1.0"
19 cachetools = "5.3.1"
20 certifi = "2023.7.22"
21 cffi = "1.16.0"
22 charset-normalizer = "3.3.0"
23 colorama = "0.4.6"
24 cryptography = "41.0.4"
25 ddt = "1.6.0"
26 exceptiongroup = "1.1.3"
27 faker = "19.11.0"
28 google-api-python-client = "2.104.0"
29 google-auth = "2.23.3"
30 google-auth-httplib2 = "0.1.1"
31 google-auth-oauthlib = "1.1.0"
32 googleapis-common-protos = "1.61.0"
33 gspread = "5.11.3"
34 h11 = "0.14.0"
35 h2 = "4.1.0"
36 hpack = "4.0.0"
37 httplib2 = "0.22.0"
38 hyperframe = "6.0.1"
39 idna = "3.4"
40 iniconfig = "2.0.0"
41 Jinja2 = "3.1.2"
42 kaitastruct = "0.10"
43 markupsafe = "2.1.3"
44 oauthlib = "3.2.2"
45 outcome = "1.3.0"
46 packaging = "23.2"
47 pandas = "2.1.1"
48 paramiko = "3.3.1"
49 pip-review = "1.3.0"
50 pluggy = "1.3.0"
51 pretty-html-table = "0.9.16"
52 protobuf = "4.24.4"
53 psycogp2 = "2.9.9"
54 py = "1.11.0"
55 pyasn1 = "0.5.0"
56 pyasn1-modules = "0.3.0"
57 pycparser = "2.21"
58 pydivert = "2.1.0"
59 pynacl = "1.5.0"
60 pyopenssl = "23.2.0"
61 pyotp = "3.1.1"
62 pyparsing = "3.1.1"
63 pysocks = "1.7.1"
64 pytest = "7.4.2"
65 pytest-html = "4.0.2"
66 pytest-metadata = "3.0.0"
67 pytest-order = "1.1.0"
68 pytest-retry = "1.5.0"
69 python-dateutil = "2.8.2"
70 python-dotenv = "1.0.0"
71 pytz = "2023.3.post1"
72 requests = "2.31.0"
73 requests-oauthlib = "1.3.1"
74 rsa = "4.9"
75 selenium = "4.14.0"
76 selenium-wire = "5.1.0"
77 six = "1.16.0"
78 sniffio = "1.3.0"
79 sortedcontainers = "2.4.0"
80 sshunnel = "0.4.0"
81 toml = "2.0.1"
82 tqdm = "4.66.1"
83 trio = "0.22.2"
84 trio-websocket = "0.11.1"
85 tldata = "2023.3"
86 uritemplate = "4.1.1"
87 urllib3 = "2.0.7"
88 webdriver-manager = "4.0.1"
89 wsproto = "1.2.0"
90 zstandard = "0.21.0"
91 numpy = "1.26.4"
92 pynput = "1.7.7"
93 opencv-python = "4.10.0.84"
94 pyautogui = "0.9.54"
95 pygetwindow = "0.0.9"
96

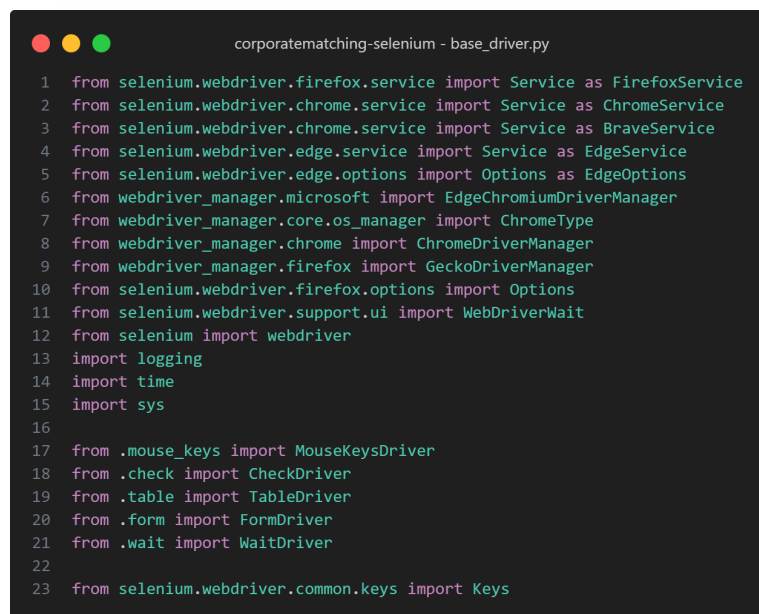
```

Gambar 3.6 Daftar Library Package Python yang Digunakan

Dalam proses ini, terdapat sekitar 85 library Python yang perlu diinstal ke dalam virtual environment Gambar 3.6, yang dapat dilakukan setelah mengaktifkan virtual environment menggunakan perintah `poetry shell`. Tahapan persiapan ini bertujuan untuk memastikan bahwa semua dependensi dan lingkungan kerja sudah siap digunakan untuk pengembangan kode otomatisasi pengujian. Kemudian terdapat dua file yang bekerja sama untuk memastikan pengujian otomatis berjalan dengan efisien dan mendukung berbagai skenario pengujian. File `Mouse Keys` menyediakan fungsi untuk mengontrol aksi seperti klik, hover, scroll, mengetik, zoom, serta mengunggah file secara otomatis

melalui Selenium. File ini juga dilengkapi dengan metode untuk pengujian fitur pagination dan interaksi elemen web yang dinamis, membantu mempercepat proses pengujian otomatis. Sedangkan file Base Driver merupakan inti dari pengaturan Selenium WebDriver untuk berbagai browser (seperti Chrome, Firefox, Edge). File ini mencakup fungsi untuk navigasi, membuka tab baru, memuat halaman, mengelola data browser, serta menyederhanakan integrasi dengan komponen tambahan seperti MouseKeysDriver. File ini mendukung pengaturan yang spesifik, baik dalam mode headless maupun fullscreen.

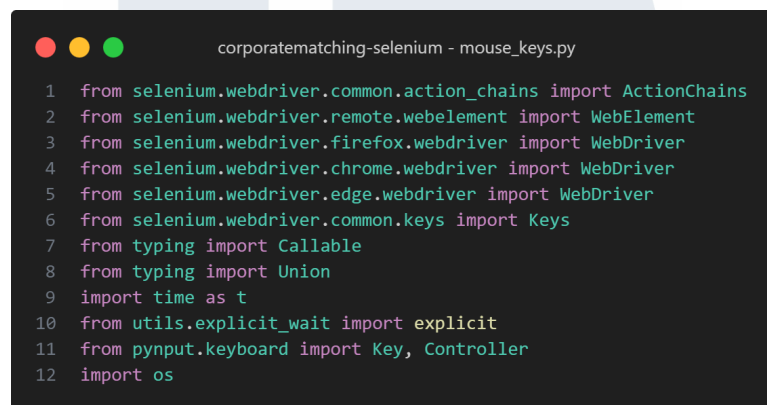
Selenium adalah sebuah pustaka (library) yang digunakan untuk otomatisasi pengujian aplikasi berbasis web [21]. Selenium memungkinkan pengembang untuk membuat skrip otomatis yang dapat mensimulasikan interaksi pengguna pada antarmuka website, seperti klik, input teks, navigasi halaman, dan pengambilan data [22]s. Selenium mendukung berbagai browser populer seperti Chrome, Firefox, dan Edge, serta menyediakan API untuk menjalankan pengujian pada platform yang berbeda. Salah satu keunggulan Selenium adalah fleksibilitasnya yang dapat digunakan bersama bahasa pemrograman populer seperti Python, Java, dan C# [21].



```
corporatematching-selenium - base_driver.py
1 from selenium.webdriver.firefox.service import Service as FirefoxService
2 from selenium.webdriver.chrome.service import Service as ChromeService
3 from selenium.webdriver.chrome.service import Service as BraveService
4 from selenium.webdriver.edge.service import Service as EdgeService
5 from selenium.webdriver.edge.options import Options as EdgeOptions
6 from webdriver_manager.microsoft import EdgeChromiumDriverManager
7 from webdriver_manager.core.os_manager import ChromeType
8 from webdriver_manager.chrome import ChromeDriverManager
9 from webdriver_manager.firefox import GeckoDriverManager
10 from selenium.webdriver.firefox.options import Options
11 from selenium.webdriver.support.ui import WebDriverWait
12 from selenium import webdriver
13 import logging
14 import time
15 import sys
16
17 from .mouse_keys import MouseKeysDriver
18 from .check import CheckDriver
19 from .table import TableDriver
20 from .form import FormDriver
21 from .wait import WaitDriver
22
23 from selenium.webdriver.common.keys import Keys
```

Gambar 3.7 Modul yang Digunakan di Base Driver

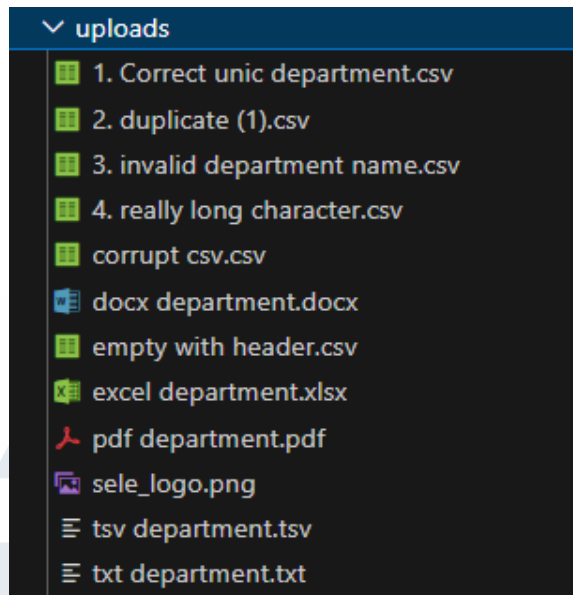
Dalam penggunaan Selenium untuk otomatisasi dengan StateSXT, beberapa library yang diimpor memiliki peran spesifik dalam menjalankan dan mengelola pengujian. Library seperti webdriver digunakan untuk mengontrol browser, sedangkan modul seperti ActionChains memungkinkan simulasi aksi yang kompleks, seperti drag-and-drop atau klik kombinasi antar fitur. Selenium juga menyediakan alat untuk menunggu eksplisit (WebDriverWait), yang memastikan elemen tertentu pada halaman telah dimuat sebelum aksi dilakukan. Terlebih karena Hashigake dikembangkan menggunakan Node.js yang membuatnya sering melakukan pemuatan web berulang.

A screenshot of a terminal window with a dark background. The title bar at the top reads 'corporatematching-selenium - mouse_keys.py'. The terminal displays 12 lines of Python code, which are imports for Selenium, typing, and other modules. The code is as follows:

```
1 from selenium.webdriver.common.action_chains import ActionChains
2 from selenium.webdriver.remote.webelement import WebElement
3 from selenium.webdriver.firefox.webdriver import WebDriver
4 from selenium.webdriver.chrome.webdriver import WebDriver
5 from selenium.webdriver.edge.webdriver import WebDriver
6 from selenium.webdriver.common.keys import Keys
7 from typing import Callable
8 from typing import Union
9 import time as t
10 from utils.explicit_wait import explicit
11 from pynput.keyboard import Key, Controller
12 import os
```

Gambar 3.8 Modul yang Digunakan di Mouse Keys Driver

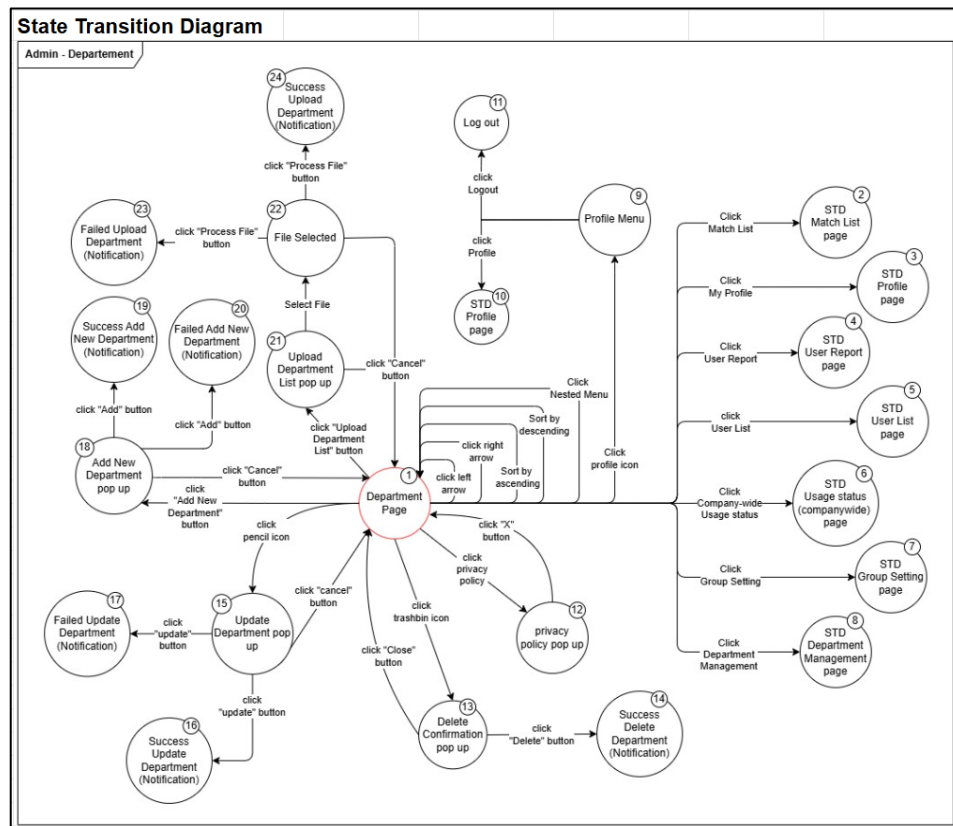
Selain itu, file `base_driver.py` (Gambar 3.7) mengatur konfigurasi dasar seperti pengaturan browser yang akan digunakan (Chrome, Firefox, Edge, atau Brave) melalui library tambahan seperti `webdriver_manager`. File ini juga mencakup modul tambahan untuk mendukung navigasi elemen kompleks dan pengelolaan kondisi unik di setiap browser. Sedangkan file `mouse_keys.py` (Gambar 3.8) digunakan untuk memanipulasi elemen berbasis keyboard dan mouse secara langsung, dengan bantuan pustaka seperti `ActionChains` dan `pynput.keyboard` untuk simulasi aksi pengguna lebih lanjut. Kombinasi kedua file ini menunjukkan bagaimana Selenium dapat diintegrasikan dengan framework lain untuk menciptakan pengujian otomatis yang lebih canggih, termasuk pengendalian browser, simulasi aksi pengguna, dan penanganan elemen dinamis pada Hashigake.



Gambar 3.9 File dan Dokumen yang Diperlukan Saat Pengujian

Untuk mendukung proses pengujian yang lebih terorganisir, framework testing dirancang agar mampu menyimpan semua file yang diperlukan dalam direktori khusus bernama uploads (Gambar 3.9). Pendekatan ini mempermudah pengujian, terutama jika dilakukan oleh orang yang berbeda, karena semua file yang dibutuhkan tersedia dalam lokasi yang jelas dan terstruktur. Proses unggah file difasilitasi oleh fungsi `upload_file`, yang secara otomatis mengambil file dari direktori uploads berdasarkan nama file yang diberikan. Fungsi ini memastikan file yang diunggah berada di lokasi yang tepat dan memberikan peringatan jika file tidak ditemukan. Dengan implementasi ini, pengujian elemen pada aplikasi dapat dilakukan dengan lebih konsisten dan efisien dan meminimalkan kesalahan akibat kelalaian dalam menyiapkan file.

Salah satu contoh fitur Hashigake yang akan dibahas adalah pengujian otomatis pada lembar test case 2.4 Admin Department, yang mencakup pengaturan departemen dalam perusahaan klien. Selain itu, sebanyak 19 halaman (pages) berhasil diotomatisasi, mencerminkan kemajuan yang signifikan dalam implementasi otomatisasi pengujian pada proyek ini. Gambar 3.10 adalah STD yang menggambarkan aliran aksi daftar skenario pengujian test case (Gambar 3.11).



Gambar 3.10 State Transition Diagram Test Scenario 2.4 Admin Department

Test Scenario					
No	State 1	State 2	State 3	State 4	Details
1	1	18	19		Success add department
2	1	18	1		Cancel add department
3	1	18	20		Failed add department
4	1	15	16		Success update department
5	1	15	1		failed update department
6	1	15	17		cancel update department
7	1	21	1		Cancel upload csv
8	1	21	22	1	Select csv then cancel upload
9	1	21	22	23	success upload csv
10	1	21	22	24	failed upload csv
11	1	13	1		Cancel delete department
12	1	13	14		Delete department
13	1	12	1		Check privacy
14	1	9	10		Change from department to profile page
15	1	9	11		Logout
16	1	2			view match list page
17	1	3			view user own profile page
18	1	4			view user report page
19	1	5			view admin - user list page
20	1	6			view admin - usage status page
21	1	7			view admin - group setting page
22	1	8			view admin - holiday management page
23	1	1			Pagination
24	1	1			Dropdown side bar
25	1	1			Sort

Gambar 3.11 Tabel Test Scenario 2.4 Admin Department

Tabel Test Scenario, setiap skenario pengujian melalui. Sebagai contoh, saat berada di halaman 1, pindah antar state, seperti State 1 (Department Page) hingga State 19 (Success Pop-Up) hingga State 20 (Success Page), untuk memastikan setiap langkah dari proses pendaftaran berhasil sesuai dengan ekspektasi. Tabel ini juga mencatat hasil pengujian, yang mencakup status pengujian, apakah sukses, gagal, atau pembatalan, yang memandu proses pengembangan.

e		Should be checked			
		Elements	Appearance	Responsiveness	Hoverable
Profile Icon	☒	☒	☒	☒	☒
Pencil Icon	☒	☒	☒	☒	☒
Trash Bin Icon	☒	☒	☒	☒	☒
Add Department Button	☒	☒	☒	☒	☒
Next Pagination Button	☒	☒	☒	☒	☒
Prev Pagination Button	☒	☒	☒	☒	☒
Department Table	☒	☒	☒	☒	☒
All menu in Left side bar	☒	☒	☒	☒	☒
Upload Department CSV Button	☒	☒	☒	☒	☒
"Add" Button	☒	☒	☒	☒	☒
"Cancel" button	☒	☒	☒	☒	☒
"Upload File" Button	☒	☒	☒	☒	☒
Department text field	☒	☒	☒	☒	☒

bar 3.12 Tabel Appearance Check 2.4 Admin D

pengecekan tampilan (appearance) UI pada hal tampak pada Gambar 3.12. Elemen-elemen s tombol tambah, tombol unggah, dan tombol- dicek keberadaannya sesuai dengan desain dapat pengecekan UI, kondisi pengujian membu k memastikan bahwa elemen-elemen website b

MULTIMEDIA

pengecekan tampilan (appearance) UI pada halaman yang akan diuji dapat dilihat pada Gambar 3.12. Elemen-elemen yang akan diuji, yaitu tombol tambah, tombol unggah, dan tombol hapus. Untuk memastikan keberadaan elemen-elemen tersebut sesuai dengan desain yang telah dibuat, dapat dilakukan pengecekan UI, kondisi pengujian membutuhkan waktu yang relatif singkat untuk memastikan bahwa elemen-elemen website berfungsi dengan baik.

yang benar dan tampil tanpa berantakan. Oleh karena itu, kode Selenium diatur hanya untuk memeriksa apakah elemen-elemen website muncul dan dapat digunakan sesuai kondisi yang diharapkan. Hal ini mencakup pemeriksaan terhadap atribut seperti responsivitas, klikabilitas, dan pengisian otomatis untuk setiap elemen.

Scenario 1								
Test Case	Domain	State 1	State 2		State 3	Expected Behavior	External Check	Note
		Department Page Add New Department Button	Add New Department Pop Up	Add Button	Success Notification			
1	Microsoft non primal	Click	Hiragana words	Click	Shown	The department will be added to the list.		
2	Microsoft non primal	Click	Katakana words	Click	Shown			
3	Microsoft non primal	Click	Kanji words	Click	Shown			
4	Microsoft non primal	Click	Latin words	Click	Shown			
5	Microsoft non primal	Click	Long words	Click	Shown			Max chars is 200, user can't type anymore character after reaching the limit
6	Microsoft Primal	Click	Hiragana words	Click	Shown	The department will be added to the list.		
7	Microsoft Primal	Click	Katakana words	Click	Shown			
8	Microsoft Primal	Click	Kanji words	Click	Shown			
9	Microsoft Primal	Click	Latin words	Click	Shown			
10	Microsoft Primal	Click	Long words	Click	Shown	The department will be added to the list.		Max chars is 200, user can't type anymore character after reaching the limit
11	Google free	Click	Hiragana words	Click	Shown			
12	Google free	Click	Katakana words	Click	Shown			
13	Google free	Click	Kanji words	Click	Shown			
14	Google free	Click	Latin words	Click	Shown			
15	Google free	Click	Long words	Click	Shown	The department will be added to the list.		Max chars is 200, user can't type anymore character after reaching the limit
16	Google paid	Click	Hiragana words	Click	Shown			
17	Google paid	Click	Katakana words	Click	Shown			
18	Google paid	Click	Kanji words	Click	Shown			
19	Google paid	Click	Latin words	Click	Shown			
20	Google paid	Click	Long words	Click	Shown			Max chars is 200, user can't type anymore character after reaching the limit

Gambar 3.13 Tabel Scenario 1 pada 2.4 Admin Department

State Transition Diagram (STD) pada halaman Department Management menggambarkan alur aksi dari elemen-elemen pada fitur ini secara sistematis, mulai dari membuka halaman, menambahkan departemen baru, hingga mendapatkan notifikasi sukses. Berdasarkan STD, skenario yang diuji dalam tabel Scenario 1 di atas melibatkan tiga langkah utama:

1. State 1: QA memulai dengan membuka halaman Department Page, di mana tombol Add New Department harus terlihat dan dapat diklik.
2. State 2: Setelah tombol tersebut diklik, Add New Department Pop Up akan muncul, memungkinkan pengguna untuk mengisi detail departemen baru.
3. State 3: Tombol Add ditekan untuk menambahkan departemen, dan sistem harus menunjukkan notifikasi sukses ("Success Notification") jika departemen berhasil ditambahkan.

Gambar 3.13 menunjukkan berbagai kombinasi input yang diuji, seperti penggunaan kata dalam Hiragana, Katakana, Kanji, Latin, dan karakter panjang

untuk nama departemen. Setiap langkah dicek untuk memastikan bahwa elemen-elemen seperti tombol atau notifikasi muncul sesuai kondisi dan berfungsi sebagaimana mestinya. Jika input valid, maka hasilnya adalah notifikasi sukses dan departemen ditambahkan ke daftar. Selain itu, catatan seperti "Max chars is 200, user can't type anymore character after reaching the limit" memastikan validasi batas karakter berjalan sesuai spesifikasi. Eksternal check memastikan bahwa hasil dari setiap langkah sesuai dengan ekspektasi, baik dari sisi fungsionalitas maupun UI. Tabel ini membantu QA mengidentifikasi kemungkinan kesalahan atau anomali pada fitur Department.

Kelas TopMenuPage dalam skrip yang diberikan mendefinisikan serangkaian metode dan fungsi untuk mengotomatisasi berbagai interaksi pengguna pada bagian menu atas aplikasi website Hashigake. Salah satu fungsi utama yang tersedia adalah autentikasi melalui metode login menggunakan Microsoft dan Google. Proses ini mencakup pengisian kredensial secara otomatis, menangani skenario seperti otentikasi dua faktor (2FA), dan popup "Stay Signed In" untuk memastikan proses login berjalan lancar. Selain itu, kelas ini juga dilengkapi metode untuk menavigasi kembali ke halaman awal (backToStart) dengan memastikan bahwa browser diarahkan ke lingkungan Dev atau Test yang sesuai berdasarkan domain yang ditentukan.

Kelas ini juga mencakup validasi domain saat ini melalui metode checkCurrentDomain, yang memastikan transisi antar domain berjalan tanpa masalah dengan melakukan logout dari domain sebelumnya jika diperlukan. Fitur login dan logout dikelola dengan metode login dan logout, yang secara otomatis menangani proses masuk dan keluar dari aplikasi berdasarkan peran pengguna, seperti "admin," "normal," "super admin," atau "unregistered." Selain itu, proses registrasi akun untuk platform Microsoft dan Google juga diotomatisasi menggunakan metode register, termasuk penanganan berbagai skenario seperti pemilihan akun yang berbeda dan pengisian detail registrasi. Gambar 3.14 adalah method Login yang diterapkan hampir di semua test case. Akun Hashigake saat ini dibagi menjadi Microsoft Free (CAC), Microsoft

Primal (klien), Google Free, dan Google Paid (admin). Update release 21 Januari menambah daftar akun yang perlu dimasukkan ke database dengan custom email berbasis utama Yahoo dan Outlook.

```

corporatematching-selenium - page.py
162 def login(self, domain, role: Literal["unregistered", "normal", "admin", "super admin"], withLogout=True, forceLogin=False, isFirst=False):
163     print(f"Running login process")
164
165     # check whether the domain changing or not
166     if self.checkCurrentDomain(domain=domain) and not forceLogin:
167         return
168
169     # exit from old domain
170     if self.isLogout and withLogout:
171         self.logout()
172
173     # login into new domain
174     print(f"Running login")
175     try:
176
177         allowedRoles = ["unregistered", "normal", "admin", "super admin"]
178         if role not in allowedRoles:
179             raise ValueError(f"Invalid role: {role}. Role must be one of {allowedRoles}")
180
181         accounts = json.load(open("../database/mocks/mock_login_valid.json"))
182         domain = domain.lower() if domain else ""
183         accountType = ""
184
185         if "microsoft" in domain:
186             accountType = "microsoft"
187         elif "microsoft primal" in domain:
188             accountType = "microsoft_primal"
189         elif "google free" in domain:
190             accountType = "google-free"
191         elif "google paid" in domain:
192             accountType = "google_paid"
193         else:
194             raise ValueError("Account does not found in database")
195
196         username = accounts[role][accountType]["username"]
197         password = accounts[role][accountType]["password"]
198
199         print(f"username: {username}")
200         print(f"password: {password}")
201
202         if "microsoft" in domain:
203             explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_SIGNIN_MICROSOFT(), useJS=True), timeout=30, raiseError=True)
204
205         def clickUserAnotherAccount():
206             self.bd.mkd.hovering(self.lr.BUTTON_USE_ANOTHER_ACCOUNT())
207             self.bd.mkd.clicking(self.lr.BUTTON_USE_ANOTHER_ACCOUNT())
208
209         isPickAccount = explicit(lambda: self.lr.TEXT_PICK_AN_ACCOUNT(), withReturn=True)
210         if isPickAccount:
211
212             explicit(clickUserAnotherAccount, timeout=15, raiseError=True)
213
214         def fillEmail():
215             self.bd.fd.insert_to_textbox(self.lr.INPUT_EMAIL(), "")
216             self.bd.fd.insert_to_textbox(self.lr.INPUT_EMAIL(), username, byEnter=True)
217
218             explicit(fillEmail, timeout=35, raiseError=True)
219
220         def fillPassword():
221             self.bd.fd.insert_to_textbox(self.lr.INPUT_PASSWORD(), password, byEnter=True)
222             self.lr.NO_TEXT_ENTER_PASSWORD()
223
224             explicit(fillPassword, timeout=25, raiseError=True)
225
226         is2fa = explicit(lambda: self.lr.TEXT_2FA(), timeout=3, withReturn=True)
227         if is2fa:
228             explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_USE_DIFFERENT_ACCOUNT()), timeout=10)
229             explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_USE_ANOTHER_ACCOUNT()), timeout=15, raiseError=True)
230             explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_EMAIL(), ""), timeout=15, raiseError=True)
231             explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_EMAIL(), username, byEnter=True), timeout=15, raiseError=True)
232             explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_PASSWORD(), password, byEnter=True), timeout=15, raiseError=True)
233             explicit(lambda: self.lr.NO_TEXT_ENTER_PASSWORD())
234
235         isStaySigned = explicit(lambda: self.lr.TEXT_STAY_SIGNED(), timeout=3, withReturn=True)
236         if isStaySigned:
237
238             def clickNoStaySignIn():
239                 self.bd.mkd.clicking(self.lr.BUTTON_NO_STAY_SIGNIN())
240                 self.lr.NO_TEXT_STAY_SIGNED()
241
242             explicit(clickNoStaySignIn, timeout=15, raiseError=True)
243
244         else:
245             explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_SIGNIN_GOOGLE(), useJS=True), timeout=30, raiseError=True)
246             accountOptions = explicit(lambda: self.lr.ACCOUNT_OPTIONS_GOOGLE(), withReturn=True)
247             if accountOptions:
248
249                 def clickAnotherAccount():
250                     self.bd.mkd.hovering(accountOptions[-1])
251                     self.bd.mkd.clicking(accountOptions[-1])
252                     self.lr.INPUT_EMAIL_GOOGLE()
253
254                 explicit(clickAnotherAccount, timeout=15, raiseError=True)
255
256             def fillEmailGoogle():
257                 self.bd.fd.insert_to_textbox(self.lr.INPUT_EMAIL_GOOGLE(), "")
258                 self.bd.fd.insert_to_textbox(self.lr.INPUT_EMAIL_GOOGLE(), username, byEnter=True)
259
260             explicit(fillEmailGoogle, timeout=15, raiseError=True)
261
262             explicit(lambda: self.bd.mkd.hovering(self.lr.INPUT_PASSWORD_GOOGLE()), timeout=15)
263             explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_PASSWORD_GOOGLE(), password, byEnter=True), timeout=15, raiseError=True)
264
265             explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_CONTINUE_GOOGLE()), timeout=6)
266
267             google2fa = explicit(lambda: self.lr.BUTTON_MORE_WAYS_TO_VERIFY(), timeout=10, withReturn=True)
268             if google2fa:
269                 raise ValueError(f"google 2fa")
270
271     except Exception as e:
272         print(f"login error: {e}\nExiting the program...")
273         self.bd.exit()
274         os._exit(1)
275

```

Gambar 3.14 Code Metode Fungsi Login di Top Menu Page

Untuk mendukung alur onboarding, metode `scrollAndAccept` digunakan untuk menggulir dan menerima kebijakan privasi secara otomatis. Metode tambahan seperti `skipRegister1`, `skipRegister2`, dan `skipRegister3` memungkinkan pengabaian langkah-langkah opsional dalam proses registrasi, seperti pengisian departemen, hobi, serta topik pembicaraan. Kelas ini juga menyediakan metode abstrak seperti `changeState` dan `resetState`, yang dirancang untuk menangani transisi status menu atas secara fleksibel. Dengan menggunakan kemampuan `StateSXT` untuk menangani eksplisit wait, interaksi elemen, dan input otomatis, kelas ini dirancang untuk memastikan alur otomatisasi yang andal, modular, dan mudah dipelihara.

A screenshot of a code editor window titled "corporatematching-selenium - test_2_4.py". The code is a Python test class named `TestAdminDepartmentManagement01` that inherits from `TestAdminDepartmentManagement`. It includes fixtures for "admin" and "usecase1". The main test method is `test_scenario001`, which performs a sequence of actions: login as admin, click department management, remove a new department, click the "add new department" button (with an assertion), insert a department name and click the "add" button (with an assertion for "success"), and finally remove the new department and logout. The code uses `self.tmp` for page objects and `self.p` for page actions.

```
6 @pytest.mark.usefixtures("admin")
7 @pytest.mark.usefixtures("usecase1")
8 class TestAdminDepartmentManagement01(TestAdminDepartmentManagement):
9
10     def __init__(self, methodName: str = "runTest"):
11         super().__init__(methodName)
12
13     @Wrapper.result_receiving
14     @Wrapper.unpagshe(*("2.4", "_SN_2_4_Scenario_001_Data"))
15     def test_scenario001(self, *args):
16         """Test Scenario: 1-18-19"""
17         try:
18             # preparation
19             self.tmp.login(domain=args[1], role="admin")
20             self.p.clickDepartmentManagement()
21             self.p.removeNewDepartment()
22
23             # click add new department button (1-18)
24             self.soft_assert(
25                 self.assertTrue,
26                 self.p.clickAddNewDepartment(isClick=args[2]),
27             )
28
29             # insert department name and click add button (18-19)
30             # success to add
31             self.soft_assert(
32                 self.assertTrue,
33                 self.p.clickAdd(words=args[3], expected="success"))
34
35             self.p.removeNewDepartment()
36         finally:
37             self.tmp.logout()
38             self.assert_all()
```

Gambar 3.15 Code Test Scenario 1 pada 2.4 Admin Department

Dalam pengembangan otomatisasi pengujian pada halaman Admin Department Management (2.4), file `test_2_4.py` (Gambar 3.15) berfungsi sebagai penghubung antara skenario yang telah dirancang di Google Sheets dan

logika kode Python berbasis framework StateSXT. Sistem ini memungkinkan eksekusi langsung dari test scenario yang didokumentasikan di sheet dengan menggunakan name range dan parameterisasi args sebagai penunjuk data input. Kode di atas menunjukkan bagaimana alur logika dari satu baris test scenario dijalankan. Pada bagian try hingga finally, setiap langkah test dijalankan secara berurutan sesuai dengan definisi dalam tabel skenario. Sebagai contoh:

1. Preparation:

- a. Sistem login ke halaman Department Management menggunakan domain dan role yang telah didefinisikan (args[1]). Fungsi untuk login sebelumnya sudah dibuat dalam folder Top Menu Page.
- b. Mengakses menu Department Management dari web.
- c. Membersihkan elemen Add New Department sebelumnya jika ada.

2. Eksekusi Skenario:

- a. Klik tombol Add New Department sesuai instruksi dalam data args[2]. Jika data bukan berupa strip atau kosong, sistem akan mengembalikan True yang berarti perlu dilakukan.
- b. Memasukkan nama departemen baru berdasarkan kata-kata yang telah ditentukan di parameter args[3].
- c. Memastikan respons yang diharapkan, di sini STD menyuruh agar penambahan nama department harus berhasil (expected value: "success").

3. Final Assertion and Cleanup:

- a. Sistem memastikan semua elemen berjalan sesuai logika yang diharapkan melalui fungsi soft_assert dan assert_all.
- b. Pada akhirnya, sistem melakukan *logout* untuk mengakhiri sesi.
- c. Loop pengujian hingga seluruh baris sebuah name range habis.

File Test.py menggunakan pendekatan parameterisasi yang memungkinkan pengujian beberapa skenario hanya dengan mengganti data input di Google Sheets. Framework StateSXT membantu dalam membaca data dari *name range* yang didefinisikan pada tabel test case, sehingga proses otomasi menjadi lebih efisien dan adaptif terhadap perubahan skenario. Dengan pendekatan ini, setiap baris skenario akan dieksekusi ulang hingga seluruh skenario pada sheet selesai diuji.

```

655 # 18
656 class AddNewDepartmentPopUpState(AdminDepartmentManagementInterface):
657
658     def __init__(self, base, contextPage) -> None:
659         super().__init__(base, contextPage)
660
661     def clickAdd(self):
662         """Required Process"""
663         words = self.param["words"].lower() if self.param["words"] else ""
664         expected = self.param["expected"]
665         direction = 3 if expected == "success" else 4
666         res = [None]
667
668         word_mapping = {
669             "dc": {"text": "000 Selenium Department", "action": lambda: explicit(lambda: self.bd.mkd.clicking(self.p.lr.BUTTON_NEW_DEPARTMENT_TRASHBIN()))},
670             "hiragana words": {"text": "0 しょうごじしゃつ", "action": lambda: explicit(lambda: self.bd.mkd.clicking(self.p.lr.BUTTON_HIRAGANA_DEPARTMENT_TRASHBIN()))},
671             "katakana words": {
672                 "text": "0 テクノロジーファッション",
673                 "action": lambda: explicit(lambda: self.bd.mkd.clicking(self.p.lr.BUTTON_KATAKANA_DEPARTMENT_TRASHBIN()))},
674             },
675         "kanji words": {"text": "0 情報技術部", "action": lambda: explicit(lambda: self.bd.mkd.clicking(self.p.lr.BUTTON_KANJI_DEPARTMENT_TRASHBIN()))},
676         "latin words": {"text": "0 IT Division", "action": lambda: explicit(lambda: self.bd.mkd.clicking(self.p.lr.BUTTON_LATIN_DEPARTMENT_TRASHBIN()))},
677         "long words": {
678             "text": "0 IT部門は 企業の情報システムを管理し 業務効率を向上させる役割を担っています システム開発 ネットワーク管理 データセキュリティなど 幅広い業務を行い 技術革新を推進します",
679             "action": lambda: explicit(lambda: self.bd.mkd.clicking(self.p.lr.BUTTON_LONG_DEPARTMENT_TRASHBIN()))},
680         },
681         "add existing department name": {"text": "Automation Engineers", "action": lambda: explicit(lambda: self.bd.mkd.hovering(self.p.lr.BUTTON_PROFILE_ICON()))},
682         "empty char (only space)": {"text": " ", "action": lambda: explicit(lambda: self.bd.mkd.hovering(self.p.lr.BUTTON_CANCEL()))},
683         "half-width words": {"text": "ソフトウェア", "action": lambda: explicit(lambda: self.bd.mkd.hovering(self.p.lr.BUTTON_CANCEL()))},
684         "mixed with numbers and special char": {"text": "0 IT Department", "action": lambda: explicit(lambda: self.bd.mkd.hovering(self.p.lr.BUTTON_CANCEL()))},
685     }
686
687     if words in word_mapping:
688         explicit(lambda: self.bd.fd.insert_to_textbox(self.p.INPUT_DESC(), word_mapping[words]["text"]), raiseError=True)
689
690         res = [
691             explicit(lambda: self.p.lr.TITLE_POPUP_DIALOG().is_displayed(), withReturn=True),
692             explicit(lambda: self.p.lr.BUTTON_CANCEL().is_enabled(), withReturn=True),
693             explicit(lambda: self.p.lr.BUTTON_ADD().is_enabled(), withReturn=True),
694         ]
695
696         res.append(explicit(lambda: self.bd.mkd.clicking(self.p.lr.BUTTON_ADD()), withReturn=True))
697
698         if expected == "success":
699             res.append(explicit(lambda: self.p.lr.NOTIFICATION_SUCCESSFUL_ADDITION().is_displayed(), withReturn=True))
700         elif expected == "failed":
701             res.append(
702                 explicit(
703                     lambda: self.p.lr.NOTIFICATION_FAILING_ADDITION().is_displayed()
704                     or self.p.lr.NOTIFICATION_INVALID_ADDITION().is_displayed()
705                     or self.p.lr.NOTIFICATION_EXISTING_ADDITION().is_displayed()
706                     or self.p.lr.NOTIFICATION_EMPTY_ADDITION().is_displayed(),
707                     withReturn=True,
708                 )
709             )
710             explicit(lambda: self.bd.click_outside_dialog())
711
712         if "action" in word_mapping[words]:
713             word_mapping[words]["action"]()
714
715         """Transition"""
716         dirs = {
717             3: SuccessAddNewDepartmentState(self.bd, self.p),
718             4: FailedAddNewDepartmentState(self.bd, self.p),
719         }
720         self.p.changeState(dirs[direction])
721
722     return all(res)
723

```

Gambar 3.16 Code Fungsi ClickAdd di File States

Fungsi ClickAdd dalam file state_2_4.py di atas adalah bagian dari proses otomatisasi pengujian untuk fitur "Admin Department Management" di aplikasi Hashigake. Fungsi ini bertugas untuk menangani skenario pengujian yang melibatkan penambahan departemen baru melalui antarmuka pengguna.

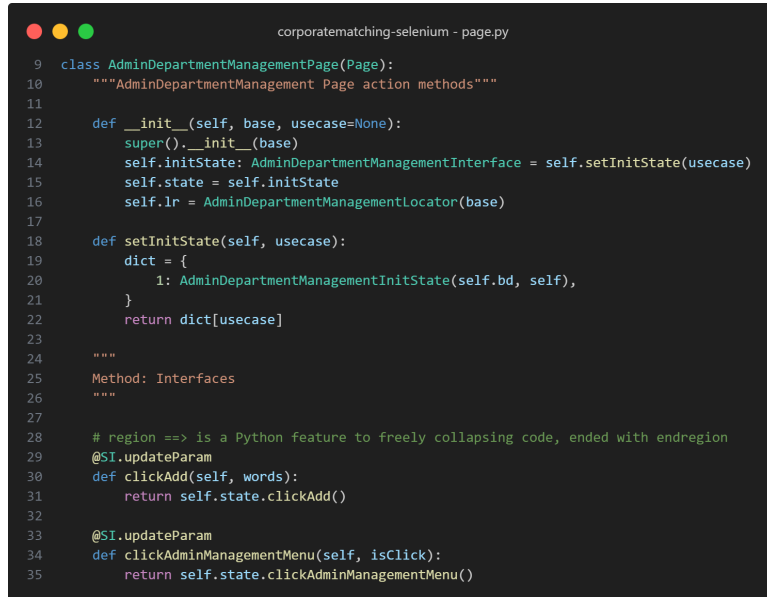
Dalam implementasinya, fungsi ini menerima parameter berupa kata atau teks yang akan dimasukkan ke dalam kolom input untuk nama departemen. Teks ini dipetakan melalui variabel `word_mapping`, yang mencakup berbagai tipe input seperti karakter Hiragana, Katakana, Kanji, Latin, serta string panjang.

Proses utama dalam fungsi ini meliputi pemetaan teks yang sesuai, memasukkan teks tersebut ke kolom input, serta memastikan elemen UI yang relevan, seperti tombol "Add" dan notifikasi keberhasilan atau kegagalan, muncul sesuai dengan logika State Transition Diagram (STD) yang telah dirancang. Selain itu, fungsi ini juga menangani kondisi di mana teks input tidak valid atau tombol "Cancel" ditekan. Dengan logika eksplisit menggunakan library `explicit_wait`, fungsi ini memastikan bahwa semua elemen UI berada pada kondisi yang tepat sebelum melanjutkan ke langkah berikutnya. Fungsi ini juga menggunakan mekanisme `res.append` untuk mencatat hasil setiap hasil aksi yang dilakukan. Variabel `Res` diharapkan mengembalikan nilai `True` atau `None` sesuai dengan yang dituliskan pada file `Test`.

Dalam konteks framework StateSXT, file `state.py` berfungsi sebagai representasi logika dan transisi status yang spesifik untuk tiap halaman atau fitur yang diuji. File ini dirancang untuk memetakan berbagai tindakan (actions) dan elemen UI berdasarkan State Transition Diagram (STD) yang telah didefinisikan. Dengan menggunakan pendekatan berbasis state, file ini membantu mengelola kompleksitas pengujian dengan cara modular dan terorganisir. Setiap metode dalam `state.py` memungkinkan integrasi mulus antara file `testcase` yang diekstrak dari Google Sheets dan tindakan otomatisasi yang dilakukan melalui Selenium, seperti yang diterapkan dalam fungsi `ClickAdd`.

Sebanyak 41 fungsi dideklarasikan dalam sistem untuk mendukung aksi pengujian seluruh STD dari fitur Admin Department Management. Gambar di atas adalah potongan kode dari file `page.py`, yang berperan sebagai penghubung antara interface pengguna dan logika otomatisasi pengujian. Pada file ini, metode seperti `clickAdd` dan `clickAdminManagementMenu` didefinisikan

untuk menjalankan aksi spesifik yang terkait dengan pengujian, seperti menambah departemen baru atau membuka menu manajemen departemen.



```

corporatematching-selenium - page.py
9 class AdminDepartmentManagementPage(Page):
10     """AdminDepartmentManagement Page action methods"""
11
12     def __init__(self, base, usecase=None):
13         super().__init__(base)
14         self.initState: AdminDepartmentManagementInterface = self.setInitState(usecase)
15         self.state = self.initState
16         self.lr = AdminDepartmentManagementLocator(base)
17
18     def setInitState(self, usecase):
19         dict = {
20             1: AdminDepartmentManagementInitState(self.bd, self),
21         }
22         return dict[usecase]
23
24     """
25     Method: Interfaces
26     """
27
28     # region ==> is a Python feature to freely collapsing code, ended with endregion
29     @SI.updateParam
30     def clickAdd(self, words):
31         return self.state.clickAdd()
32
33     @SI.updateParam
34     def clickAdminManagementMenu(self, isClick):
35         return self.state.clickAdminManagementMenu()

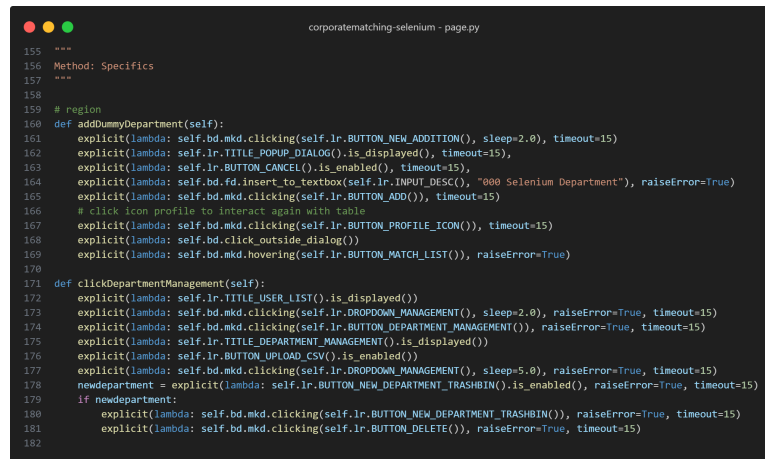
```

Gambar 3.17 Potongan Fille Page pada 2.4 Admin Department

Kelas `AdminDepartmentManagementPage` menginisialisasi state awal menggunakan `setInitState`, yang menciptakan struktur modular untuk mengelola berbagai skenario pengujian. Fungsi-fungsi ini memanfaatkan decorator `@ST.updateParam` untuk mempermudah parameterisasi, sehingga setiap aksi dapat disesuaikan berdasarkan kebutuhan pengujian. Kode ini menjadi komponen penting dalam memastikan semua elemen pada halaman diuji secara menyeluruh dan sesuai dengan alur yang dirancang pada STD.

Pada potongan kode di Gambar 3.18, fungsi-fungsi seperti `addDummyDepartment` dan `clickDepartmentManagement` ditambahkan ke halaman `page.py` untuk menangani aksi-aksi spesifik yang tidak tercantum secara langsung dalam State Transition Diagram (STD). Fungsi-fungsi ini memastikan bahwa pengujian tetap mencakup skenario penting lainnya, seperti menambahkan data dummy atau membuka halaman manajemen departemen. Setiap aksi menggunakan metode explicit untuk memastikan bahwa elemen yang diperlukan tersedia dan dapat diakses sebelum menjalankan perintah,

sehingga menjaga stabilitas dan keakuratan pengujian. Hal ini membantu melengkapi cakupan otomatisasi di luar alur standar yang ada di STD.

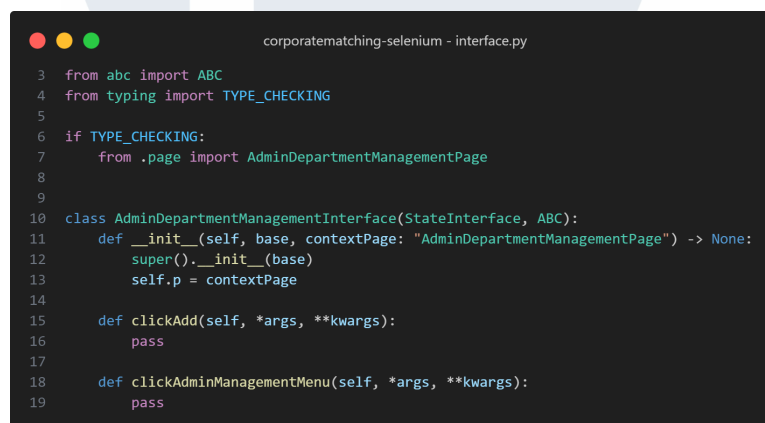


```

155
156 Method: Specifics
157
158
159 # region
160 def addDummyDepartment(self):
161     explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_NEW_ADDITION()), sleep=2.0, timeout=15)
162     explicit(lambda: self.lr.TITLE_POPUP_DIALOG().is_displayed(), timeout=15),
163     explicit(lambda: self.lr.BUTTON_CANCEL().is_enabled(), timeout=15),
164     explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_DESC(), "000 Selenium Department"), raiseError=True)
165     explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_ADD()), timeout=15)
166     # click icon profile to interact again with table
167     explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_PROFILE_ICON()), timeout=15)
168     explicit(lambda: self.bd.click_outside_dialog())
169     explicit(lambda: self.bd.mkd.hovering(self.lr.BUTTON_MATCH_LIST()), raiseError=True)
170
171 def clickDepartmentManagement(self):
172     explicit(lambda: self.lr.TITLE_USER_LIST().is_displayed())
173     explicit(lambda: self.bd.mkd.clicking(self.lr.DROPDOWN_MANAGEMENT()), sleep=2.0, raiseError=True, timeout=15)
174     explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_DEPARTMENT_MANAGEMENT()), raiseError=True, timeout=15)
175     explicit(lambda: self.lr.TITLE_DEPARTMENT_MANAGEMENT().is_displayed())
176     explicit(lambda: self.lr.BUTTON_UPLOAD_CSV().is_enabled())
177     explicit(lambda: self.bd.mkd.clicking(self.lr.DROPDOWN_MANAGEMENT()), sleep=5.0, raiseError=True, timeout=15)
178     newdepartment = explicit(lambda: self.lr.BUTTON_NEW_DEPARTMENT_TRASHBIN().is_enabled(), raiseError=True, timeout=15)
179     if newdepartment:
180         explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_NEW_DEPARTMENT_TRASHBIN()), raiseError=True, timeout=15)
181         explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_DELETE()), raiseError=True, timeout=15)
182

```

Gambar 3.18 Potongan Method Spesifics di File Page



```

3 from abc import ABC
4 from typing import TYPE_CHECKING
5
6 if TYPE_CHECKING:
7     from .page import AdminDepartmentManagementPage
8
9
10 class AdminDepartmentManagementInterface(StateInterface, ABC):
11     def __init__(self, base, contextPage: "AdminDepartmentManagementPage") -> None:
12         super().__init__(base)
13         self.p = contextPage
14
15     def clickAdd(self, *args, **kwargs):
16         pass
17
18     def clickAdminManagementMenu(self, *args, **kwargs):
19         pass

```

Gambar 3.19 Potongan File Interface pada 2.4 Admin Department

Pada file interface.py yang ditampilkan pada Gambar 3.19, kelas AdminDepartmentManagementInterface berfungsi sebagai penghubung antara STD dan implementasi halaman tertentu seperti AdminDepartmentManagementPage. Dalam konteks StatesXT, file ini mendefinisikan antarmuka yang digunakan untuk menjamin konsistensi dalam penerapan metode-metode yang dideklarasikan di setiap state. Kelas ini mengimplementasikan dua metode, yaitu clickAdd dan clickAdminManagementMenu, yang masing-masing berfungsi sebagai

placeholder untuk aksi-aksi spesifik yang akan diisi di file state.py. Pendekatan ini memungkinkan sistem untuk memisahkan logika eksekusi dari definisi struktur, mempermudah pemeliharaan dan pengembangan otomatisasi pengujian.

Selain itu, dengan menggunakan interface seperti ini, StatesXT memastikan bahwa semua aksi yang dibutuhkan oleh halaman tertentu terorganisasi dengan baik, memungkinkan integrasi yang lebih mulus antara state dan halaman. Hal ini sesuai dengan filosofi StatesXT untuk menciptakan framework pengujian yang modular dan dapat diandalkan.



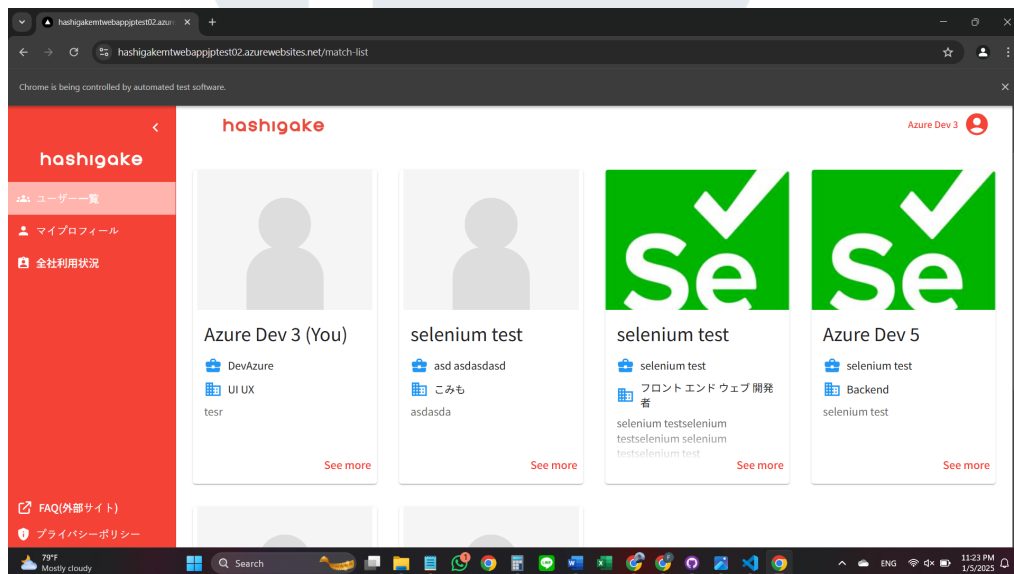
Gambar 3.20 Isi File Locator Hashigake untuk 2.4 Admin Department

File locator.py pada framework StatesXT berfungsi sebagai pusat referensi untuk semua XPath atau lokasi elemen website yang digunakan dalam otomatisasi pengujian. File ini dirancang untuk memisahkan definisi elemen-elemen dari logika program utama, sehingga membuat kode lebih modular,

terstruktur, dan mudah dipelihara. Sebagai contoh, elemen tombol "Button_Add" dapat didefinisikan seperti berikut:

```
self.BUTTON_ADD = lambda loc="//button[normalize-space()='Add']":  
    self.bd.wd.clickable(self.by.xpath, loc)
```

Kode di atas ini menunjukkan bahwa elemen tombol "Button_Add" dipilih menggunakan XPath (self.by.xpath) dan hanya akan dieksekusi jika elemen tersebut dapat diinteraksi (clickable). File locator.py mempermudah pembaruan lokasi elemen jika terjadi perubahan pada desain website, mengurangi duplikasi kode, dan memastikan semua lokasi elemen dikelola di satu tempat. Dengan pendekatan ini, framework StatesXT menjamin konsistensi dalam menemukan dan mengakses elemen-elemen website di seluruh proses pengujian otomatis.



Gambar 3.21 Proses Pengujian Saat Selenium WebDriver Berjalan

Setelah seluruh kode di dalam folder test case selesai dibuat, proses pengujian dapat dilakukan menggunakan decorator @pytest.mark.dev di atas sebuah scenario ataupun di file Init.py Scenarios untuk menjalankan semua skenario di sebuah test case. Pengujian ini akan mengeksekusi seluruh skenario yang telah didefinisikan sebelumnya, mencatat hasil status seperti PASSED

atau lainnya, dan menghitung waktu eksekusi untuk setiap skenario. Contohnya, dalam gambar terlihat bahwa setiap skenario memiliki hasil status PASSED dengan waktu eksekusi yang bervariasi, seperti 393.38 detik untuk "Test_Scenario0001". Dokumentasi hasil ini membantu QA dalam memantau performa pengujian dan memastikan tidak ada skenario yang gagal selama proses pengujian otomatis.



```

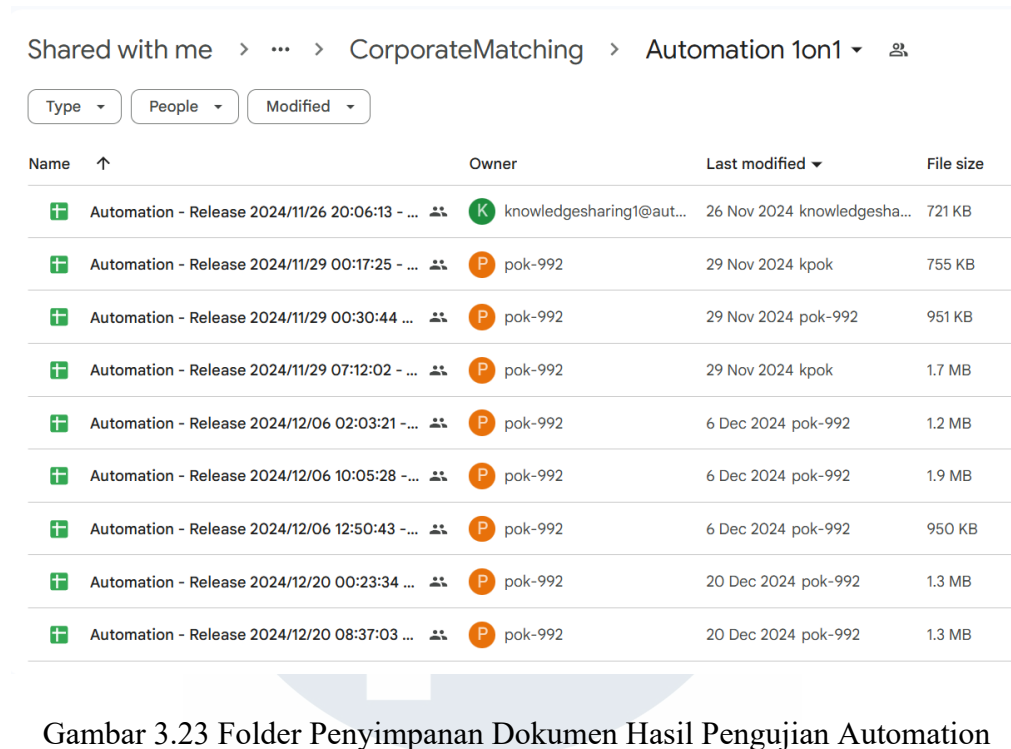
{
  "Scenario": {
    "TestUserReport01": {
      "Test Scenario001": "PASSED - Total time taken: 393.38",
      "Test Scenario002": "PASSED - Total time taken: 195.62",
      "Test Scenario003": "PASSED - Total time taken: 187.87",
      "Test Scenario004": "PASSED - Total time taken: 212.65",
      "Test Scenario005": "PASSED - Total time taken: 178.23",
      "Test Scenario006": "PASSED - Total time taken: 192.88",
      "Test Scenario007": "PASSED - Total time taken: 184.32"
    }
  },
  "Test Cases": {
    "1.4": {
      "_SN_1_4_Scenario_001_Data": [
        [
          "PASSED",
          "",
          "",
          39.18
        ],
        [
          "PASSED",
          "",
          "",
          11.6
        ],
        [
          "PASSED",
          "",
          "",
          10.95
        ]
      ]
    }
  }
}

```

Gambar 3.22 Potongan Hasil Pengujian di File Result.js

Setelah proses pengujian otomatis selesai dijalankan, file hasil pengujian, seperti result.js, akan digunakan untuk mengisi dokumen test case yang terdapat di Google Sheet. File ini mencatat hasil setiap pengujian, termasuk status seperti PASSED, catatan penguji (Tester Note) yang mana diisi oleh Automation Selenium, hasil cek eksternal (External Check Result), serta waktu eksekusi masing-masing skenario. Data dari result.js kemudian dimasukkan ke dalam format tabel di Google Sheet, seperti yang terlihat pada gambar, sehingga memudahkan tim QA untuk melakukan analisis dan

pelacakan progres pengujian secara terstruktur. Pendekatan ini memastikan hasil pengujian terdokumentasi dengan baik dan dapat diakses oleh seluruh tim.



Date	Tester	Result	External Check Result	Tester Note	Execution Time
2024/11/25 14:05:50	Automation Selenium	PASSED			45.34
2024/11/25 14:05:50	Automation Selenium	PASSED			61.97
2024/11/25 14:05:50	Automation Selenium	PASSED			40.41
2024/11/25 14:05:50	Automation Selenium	PASSED			48.9
2024/11/25 14:05:50	Automation Selenium	PASSED			81.21
2024/11/25 14:05:50	Automation Selenium	PASSED			33.64
2024/11/25 14:05:50	Automation Selenium	PASSED			32.85
2024/11/25 14:05:50	Automation Selenium	PASSED			35.52
2024/11/25 14:05:50	Automation Selenium	PASSED			41.44
2024/11/25 14:05:50	Automation Selenium	PASSED			40.4
2024/11/25 14:05:50	Automation Selenium	PASSED			35.55
2024/11/25 14:05:50	Automation Selenium	PASSED			50.13

Gambar 3.24 Contoh Tabel Hasil Pengujian Automation

```

corporatematching-selenium - page.py

224 def updateHoliday(self, inputMethod, desc):
225
226     def getPreviousDate(dateStr, dateFormat="%m/%d/%Y"):
227         dateObj = datetime.strptime(dateStr, dateFormat)
228         previousDateObj = dateObj - timedelta(days=1)
229         return previousDateObj.strftime(dateFormat)
230
231     def generateUniqueDescription(previousDesc, wordCount=8):
232         newDesc = " ".join("".join(random.choices(string.ascii_lowercase, k=random.randint(3, 8))) for _ in range(wordCount)) + "."
233
234         if previousDesc == newDesc:
235             return generateUniqueDescription(previousDesc, wordCount)
236
237         return newDesc
238
239     def getValueText():
240         dateText = explicit(lambda: self.lr.INPUT_DATE_POPUP().get_attribute("value"), withReturn=True)
241         descText = explicit(lambda: self.lr.INPUT_DESC().get_attribute("value"), withReturn=True)
242         return dateText, descText
243
244     currentDateText, currentDescText = getValueText()
245
246     if desc and "update" in desc:
247         generatedDesc = generateUniqueDescription(desc, 6)
248         explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_DESC(), generatedDesc))
249     elif not desc:
250         explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_DESC(), ""))
251
252     if not inputMethod:
253         explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_DATE_POPUP(), ""))
254         updatedDateText, updatedDescText = getValueText()
255
256         if desc and "update" in desc:
257             return updatedDateText != currentDateText and currentDescText != updatedDescText
258
259         if desc and "current" in desc:
260             return updatedDateText != currentDateText and currentDescText == updatedDescText
261
262         if not desc:
263             return updatedDateText != currentDateText and currentDescText != updatedDescText
264
265     if inputMethod == "select":
266         explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_CALENDAR()))
267         isPrevDateExist = explicit(lambda: self.lr.BUTTON_PREV_DATE().is_displayed(), withReturn=True, timeout=5)
268         if isPrevDateExist:
269             explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_PREV_DATE()))
270         else:
271             explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_NEXT_DATE()), raiseError=True)
272
273     elif inputMethod == "current":
274         explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_CALENDAR()))
275         explicit(lambda: self.bd.mkd.clicking(self.lr.BUTTON_SELECTED_DATE()))
276
277     elif inputMethod == "type":
278         previousDate = getPreviousDate(currentDateText)
279         explicit(lambda: self.bd.fd.insert_to_textbox(self.lr.INPUT_DATE_POPUP(), previousDate))
280
281     updatedDateText, updatedDescText = getValueText()
282
283     if desc and "update" in desc and (inputMethod == "select" or inputMethod == "type"):
284         return updatedDateText != currentDateText and currentDescText != updatedDescText
285
286     if desc and "current" in desc and (inputMethod == "select" or inputMethod == "type"):
287         return updatedDateText != currentDateText and currentDescText == updatedDescText
288
289     if not desc and (inputMethod == "select" or inputMethod == "type"):
290         return updatedDateText != currentDateText and updatedDescText == ""
291
292     if not desc and inputMethod == "current":
293         return updatedDateText == currentDateText and updatedDescText == ""
294
295     if desc and "update" in desc and inputMethod == "current":
296         return updatedDateText == currentDateText and currentDescText != updatedDescText
297
298     if desc and "current" in desc and inputMethod == "current":
299         return updatedDateText == currentDateText and currentDescText == updatedDescText
300
301     return False
302

```

Gambar 3.25 Fungsi Aksi pada Fitur Date Calendar Button dan Input

Kerja magang ini juga menghasilkan beragam kode otomasi aksi pada bermacam-macam elemen website Hashigake lain. Pada Gambar 3.25, terdapat fungsi menarik yang digunakan untuk melakukan aksi pada elemen kalender di halaman aplikasi, yaitu fungsi `updateHoliday`. Fungsi ini dirancang untuk

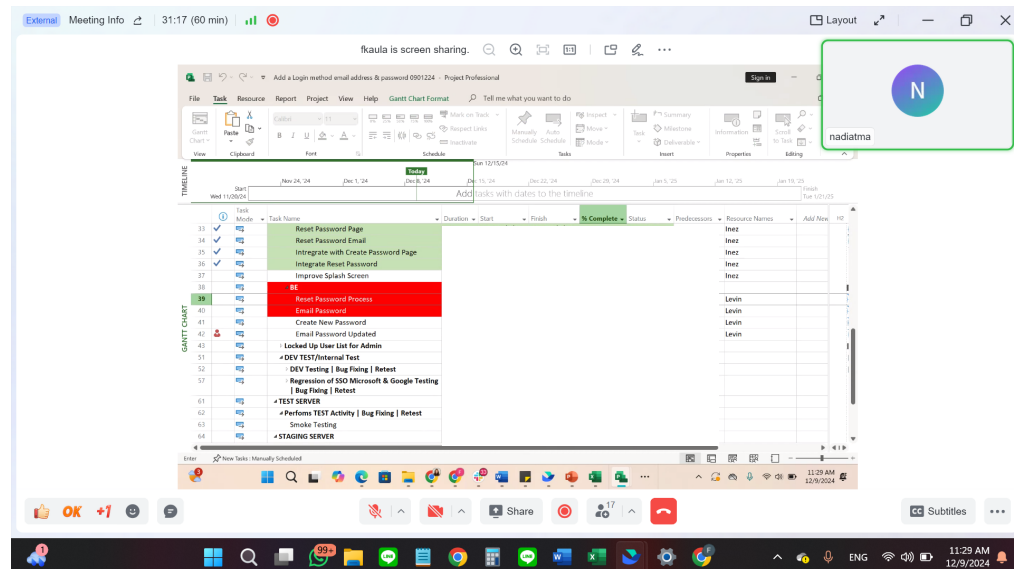
memperbarui deskripsi dan tanggal pada elemen input kalender melalui berbagai metode, seperti memilih tanggal sebelumnya, memilih tanggal saat ini, atau memasukkan tanggal secara manual. Fungsi ini memanfaatkan XPath untuk menemukan elemen kalender dan menerapkan aksi seperti klik pada tombol navigasi kalender, mengisi teks ke dalam input tanggal, serta memeriksa nilai input yang diperbarui.

Fungsi ini juga memiliki kemampuan untuk menghasilkan deskripsi unik secara otomatis melalui metode `generateUniqueDescription`, yang memastikan deskripsi tidak duplikat dengan nilai sebelumnya. Selain itu, terdapat logika validasi untuk memastikan bahwa perubahan tanggal dan deskripsi sesuai dengan input yang diberikan, termasuk memeriksa apakah perubahan berhasil diterapkan. Dengan pemanfaatan pustaka `explicit` untuk mengontrol waktu tunggu dan aksi yang presisi, fungsi ini membantu memastikan elemen kalender berfungsi dengan benar dalam berbagai skenario pengujian. Hal ini sangat relevan dalam konteks otomatisasi pengujian, di mana stabilitas dan validasi elemen kalender sangat penting untuk memastikan integritas aplikasi Hahigake.

3.2.3 Uji Fitur Baru dan Regression Test

Sesuai dengan penugasan oleh QA Manager, kerja magang ini juga bertanggung jawab membantu tim QA manual tester ketika diperlukan tambahan *resources*, khususnya saat ada perubahan besar dari sistem Hashigake. Pengujian dilakukan secara manual untuk memastikan bahwa setiap fitur baru pada aplikasi Hashigake berfungsi sesuai dengan spesifikasi yang telah ditetapkan dalam Technical Specification Document. Bersama tim QA lainnya, pengujian ini mencakup validasi fungsionalitas fitur baru serta pengujian regresi untuk memastikan bahwa fitur-fitur lama tetap berjalan dengan baik setelah adanya pembaruan atau penambahan fitur. Proses ini bertujuan untuk mengidentifikasi bug atau anomali pada sistem, sehingga aplikasi dapat memberikan pengalaman pengguna yang optimal dan konsisten.

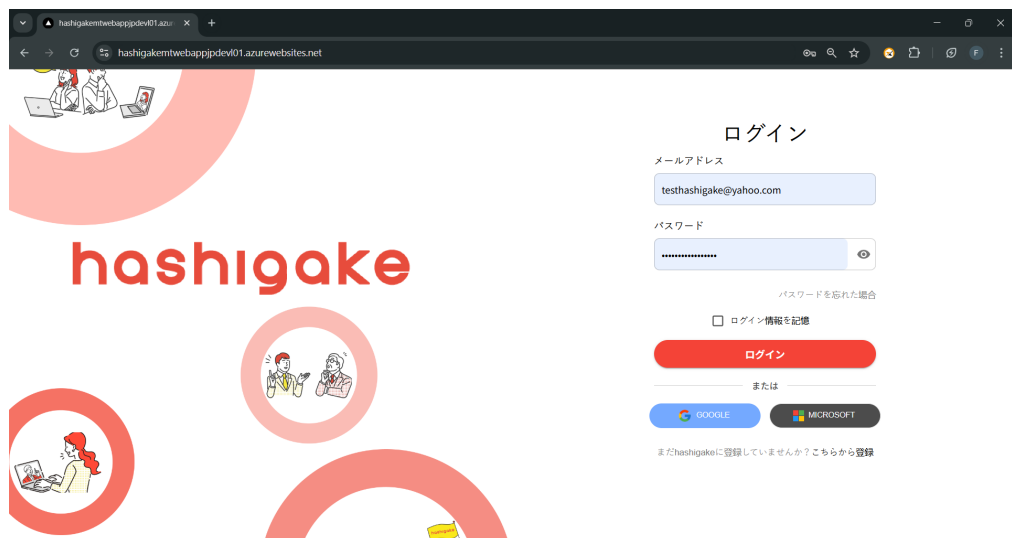
Kolaborasi dalam tim menjadi kunci utama dalam menyelesaikan pengujian dengan akurasi tinggi dan dalam waktu yang efisien.



Gambar 3.26 Pelaporan Progres Kerja Setiap Tim 1on1 Matching pada Meeting Daily Stand Up bersama Developer

Selama pengujian regression test pada bulan Desember 2024, tim QA memfokuskan pada perubahan besar berupa implementasi Custom Login dan Custom Logging API. Hasil pengujian mencatat total 85 temuan bug dengan prioritas yang beragam, terdiri dari 29 bug prioritas tinggi, 24 bug prioritas rendah, dan 32 bug prioritas menengah. Semua pengujian dilakukan melalui internal testing di DEV Server, dengan tingkat keberhasilan mencapai 99.5%.

Namun, selama proses regression test, tim QA menghadapi berbagai hambatan yang datang dari tim developer. Salah satu kendala utama adalah keterlambatan penyelesaian tugas yang telah dijanjikan oleh developer, sehingga menyebabkan blocker bagi tim QA maupun tim developer lainnya. Selain itu, kurangnya koordinasi, seperti lupa memberikan informasi apakah perubahan sudah dideploy di lingkungan tertentu atau belum disatukan ke branch utama, sering kali memperlambat proses pengujian. Kendala-kendala ini menuntut komunikasi yang lebih proaktif antar tim untuk memastikan efisiensi dalam setiap tahap pengujian.



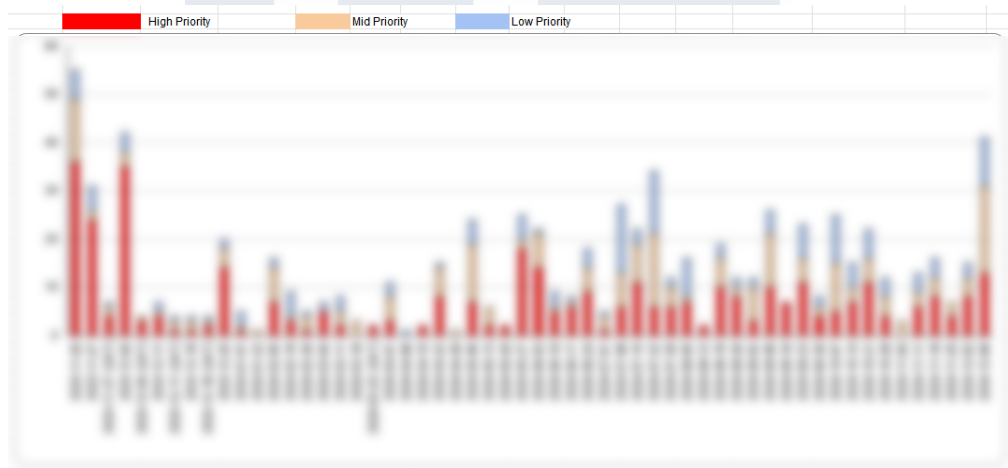
Gambar 3.27 Halaman Login pada Website Hashigake Custom Dev

Adapun komponen yang perlu segera diuji pada environment Custom Dev meliputi Login, Sign Up, Forgot Password, dan Locked Account, yang masing-masing menunjukkan tingkat keberhasilan 98%-100%. Beberapa kendala yang diidentifikasi melibatkan spam email dan tampilan UI yang rusak di email Outlook klasik serta masalah terkait meeting, tombol unduhan di laporan pengguna, dan masa berlaku login token.

Namun, Smoke Testing pada TEST Server dan STAG Server belum menunjukkan perkembangan signifikan dengan tingkat keberhasilan 0% akibat fitur ini belum dinaikkan ke environment tersebut. Hingga akhir penulisan laporan magang ini fitur sinkronisasi kalender dengan Yahoo Mail, Gmail, dan Outlook Mail belum diselesaikan yang mengakibatkan fitur Matching belum bekerja dengan sempurna. Pengujian ini memberikan wawasan penting bagi pengembangan lebih lanjut, memastikan bahwa sistem dapat berfungsi optimal setelah penerapan perubahan besar tersebut.

Seluruh bug yang ditemukan dan test case yang diuji oleh masing-masing staf QA selama periode pengujian dimasukkan ke dalam dokumen Bug Discovery Rate Graph. Dokumen ini digunakan untuk kebutuhan manajerial guna memberikan gambaran menyeluruh tentang performa pengujian dan

efektivitas deteksi bug di seluruh proyek yang dikerjakan oleh FOS, termasuk pencapaian dan kontribusi setiap anggota tim QA. Untuk proyek 1on1 Matching, grafik Bug Discovery Rate (Gambar 3.28) menunjukkan tren deteksi bug sejak awal proyek, sebagaimana terlihat pada gambar. Grafik ini menggambarkan distribusi bug berdasarkan prioritas, yaitu High Priority, Mid Priority, dan Low Priority, yang masing-masing diwakili oleh warna merah, oranye, dan biru. Data ini memberikan kredensial penting bagi manajemen untuk mengevaluasi efektivitas pengujian dan memprioritaskan perbaikan bug sesuai dengan tingkat urgensinya. Hasil ini mencerminkan kontribusi QA dalam menjaga kualitas aplikasi sesuai dengan standar yang ditetapkan oleh FOS.



Gambar 3.28 Catatan Bug Discovery Mingguan 1on1 Matching

Selama periode kerja magang ini, hasil pengujian menunjukkan bahwa sebanyak 275 test case manual berhasil dieksekusi tanpa ada pembaruan atau penambahan test case baru selama proses tersebut. Di sisi otomatisasi, 2.089 test case baru dibuat dengan 1.294 test case aktif diperbarui dan 1.092 test case otomatisasi dieksekusi berhasil dieksekusi, yang mencerminkan tingkat aktivitas tinggi dalam memastikan kualitas website Hashigake.

Pada seluruh tahapan pengujian yang dilakukan penulis, ditemukan 26 bug, terdiri dari 5 bug kategori high, 14 bug kategori mid, dan 7 bug kategori low, di mana 22 bug telah berhasil diperbaiki dalam satu periode minggu yang sama setelah bug ditemukan. Hasil ini menunjukkan efektivitas kolaborasi

dalam tim QA untuk mendeteksi dan menyelesaikan permasalahan dalam aplikasi secara tepat waktu dan sistematis. Laporan magang ini akan membahas lima temuan bug dengan label high priority karena bug ini biasanya dianggap paling kritis untuk dilaporkan kepada manajemen.

3.2.3.1 Bug ID-2 RELEASE-20250121 (Custom Login): Kesalahan Redirect pada Fitur Login/Sign-Up Menggunakan Akun Google

Bug ini ditemukan pada tanggal 10 Desember 2024 di lingkungan pengujian DEV dengan kategori prioritas High. Masalah terjadi pada fitur login dan sign-up, khususnya ketika pengguna mencoba mengakses Hashigake menggunakan akun Google. Bug ini menyebabkan pengguna diarahkan ke tautan yang salah, yaitu <https://app-dev.hashigake.jp/>, alih-alih diarahkan ke tautan <https://hashigakemtwebappjpdevl01.azurewebsites.net/> yang benar. Masalah ini hanya terjadi ketika pengguna menggunakan akun Google, sementara mekanisme login lainnya tidak terdampak.

Temuan ini didokumentasikan oleh Gempar, dan referensi pengujian dicatat melalui alat Bugproof. Tester memberikan catatan bahwa bug ini terbatas pada akun Google, dan ekspektasi hasil yang benar adalah pengguna harus diarahkan ke tautan yang sesuai saat melakukan login atau sign-up. Bug ini telah diperbaiki oleh Daniel dan ditandai dengan status Done setelah dilakukan retest oleh tim QA, yang melibatkan Nicholas, Dhea, dan Gempar pada tanggal 11 Desember 2024. Bug ini menjadi salah satu perhatian utama karena tingkat prioritasnya yang tinggi dan urgensi untuk menjaga alur pengalaman pengguna tetap lancar.

3.2.3.2 Bug ID: 48 RELEASE-20250121 (Custom Login): Sign-Up Issue via SSO Google

Bug ini ditemukan pada tanggal 17 Desember 2024 di lingkungan pengujian DEV dan memiliki tingkat prioritas High. Masalah ini memungkinkan pengguna untuk mengakses match list dengan profil kosong jika mereka menggunakan tautan langsung melalui URL tanpa menyelesaikan

proses registrasi. Kasus ini terjadi ketika pengguna menggunakan akun Google gratis dan melakukan sign-up melalui SSO (Single Sign-On).

Ekspektasi hasil yang diharapkan adalah pengguna yang mencoba mengakses match list tanpa melengkapi proses registrasi akan diarahkan kembali ke halaman Login Page. Bug ini ditemukan oleh Gempar dan didokumentasikan melalui Bugproof.

Setelah diperbaiki oleh Vincent, pengujian ulang dilakukan pada tanggal 19 Desember 2024 oleh Gempar, dengan hasil Passed in Custom DEV. Namun, catatan menyebutkan bahwa perlu koordinasi lebih lanjut dengan Gempar untuk memastikan pengujian pada skenario ini selesai sepenuhnya. Bug ini kini telah ditandai dengan status Done.

3.2.3.3 Bug ID: 52 RELEASE-20250121 (Custom Login): User Pre-Register Error with Long Text

Bug ini ditemukan pada tanggal 19 Desember 2024 di lingkungan DEV dengan tingkat prioritas High. Masalah terjadi saat pengguna memasukkan nama dengan teks panjang yang mencapai batas tertentu, muncul pesan kesalahan "The input data is not a complete block." Selain itu, pengguna baru yang dimasukkan tidak tercatat dalam tabel. Hal ini juga berdampak pada tombol Next/Prev di tabel daftar pengguna, di mana kesalahan serupa dapat terjadi secara acak, menyebabkan QA tidak dapat mengakses lebih dari 5 pengguna dalam daftar. Hasil yang diharapkan adalah pengguna dapat melakukan pre-register tanpa adanya notifikasi kesalahan, dan data pengguna ditampilkan dengan benar dalam tabel. Bug ini ditemukan oleh Gempar dan didokumentasikan melalui Bugproof.

Perbaikan dilakukan oleh Daniel, dengan pengujian ulang oleh Gempar pada tanggal 23 Desember 2024. Hasil pengujian ulang menunjukkan bahwa masalah telah Passed in Custom DEV dan Passed in DEV. Namun, pengembang mencatat bahwa masalah ini disebabkan oleh data type issue pada prosedur penyimpanan. Ketika nama lengkap dengan panjang 200 karakter

dienkripsi, panjangnya melebihi batas 500 karakter yang didukung oleh prosedur penyimpanan. Hal ini menyebabkan teks terenkripsi terpotong, dan saat didekripsi, terjadi kesalahan dengan pesan "The input data is not a complete block." Bug ini telah diselesaikan dan ditandai dengan status Done.

3.2.3.4 Bug ID: 65 RELEASE-20241125 (ROPF RUFG): Status Meeting Incorrectly Marked as "No Show"

Bug ini ditemukan pada tanggal 9 Desember 2024 di lingkungan DEV dengan tingkat prioritas *High*. Masalah terjadi pada fitur *Status Meeting*, di mana saat pengguna melakukan *force matching meeting* menggunakan akun tes (Dev1 dan Dev9), status pertemuan yang seharusnya "Finished" berubah menjadi "No Show" meskipun peserta telah hadir dan melakukan pertemuan.

Catatan Pengujian:

- Bug ini muncul ketika menggunakan mode *Edge Incognito*.
- Pengguna merefresh halaman *meeting*.
- Peserta lain meninggalkan pertemuan saat pengguna masih dalam proses refresh.

Detail Pengguna:

- Dev1 dan Dev9 FOS (11:21 - 11:41 JST).
- Dev1, Dev2, dan Dev3 ATA (11:15 - 11:45 JST).

Hasil yang Diharapkan:

Status pertemuan akan secara akurat menampilkan "Finished" setelah pertemuan selesai, sesuai dengan kehadiran peserta. Bug ini ditemukan oleh Gempar dan didokumentasikan melalui Bugproof. Tugas perbaikan diberikan kepada Levin, dengan retest dilakukan oleh Jason dan Rahma pada berbagai tanggal, yaitu 11, 12, 13, dan 16 Desember 2024. Hasil pengujian ulang menunjukkan bahwa bug ini telah *Passed on DEV, TEST, dan STAG*. Status perbaikan bug ini dinyatakan *Done*.

3.2.3.5 Bug ID: 18 RELEASE-20240904: *Matching Feedback Report Page* Forbidden Error

Bug ini ditemukan pada tanggal 23 Agustus 2024 di lingkungan DEV dengan tingkat prioritas *High*. Masalah terjadi pada halaman *Matching Feedback Report Page* yang muncul di menu sidebar di bawah menu *User Report*, tetapi tidak dapat diakses dan memunculkan pesan *forbidden error*.

Langkah Reproduksi:

1. Masuk (*sign in*) menggunakan akun *normal user*.
2. Halaman *hashigake* muncul.
3. Keluar (*sign out*).
4. Masuk kembali menggunakan akun *admin*.
5. Halaman *hashigake* muncul.
6. Klik daftar bersarang (*nested list*) pada menu *admin report*.
7. Keluar (*sign out*).

Hasil yang Diharapkan:

1. Untuk *normal user*: Menu *superadmin* tidak seharusnya muncul di menu sidebar pengguna normal.
2. Untuk *admin*: Menu *superadmin* seharusnya berada di dalam *nested list* laporan (*report nested list*), bukan di bawah menu *User Report*.

Bug ini ditemukan oleh Gempar dan dicatat menggunakan Bugproof. Perbaikan bug selesai pada tanggal 27 Agustus 2024 dan diuji ulang oleh Pok. Hasil pengujian ulang menunjukkan bahwa bug ini telah *Passed in DEV*. Penanggung jawab perbaikan adalah Elbert, dan status bug dinyatakan *Done*.

3.3 Kendala yang Ditemukan

3.3.1 Ketidakpahaman terhadap Prioritas Bug

Salah satu tantangan utama adalah memahami sistem prioritas bug yang diterapkan dalam proyek. Prioritas bug yang terdiri dari kategori High, Medium, dan Low seringkali sulit untuk diidentifikasi dengan jelas, terutama bagi anggota tim yang belum memiliki pengalaman mendalam dalam pengelolaan bug. Kendala ini berpengaruh pada proses pelaporan dan pengambilan keputusan terkait bug mana yang harus didahulukan untuk diperbaiki. Selain itu, kurangnya pemahaman mengenai dampak bug pada sistem secara keseluruhan juga menambah kompleksitas dalam menentukan prioritas.

3.3.2 Perubahan Mayor pada Hashigake Selama Proses Automation

Tantangan lain yang signifikan adalah adanya perubahan mayor pada aplikasi Hashigake di tengah proses otomatisasi pengujian. Perubahan ini mencakup pembaruan besar pada fitur-fitur inti, seperti Features Merger, Features Removal, Custom Login dan Custom Logging API. Perubahan yang tidak terduga ini sering menyebabkan kegagalan dalam skrip otomatisasi yang telah dibuat sebelumnya, sehingga memerlukan modifikasi berulang pada kode otomatisasi yang sangat memakan banyak alokasi sumber daya. Selain itu, perubahan tersebut juga menuntut QA untuk melakukan retesting dan regression testing secara berkala, yang pada akhirnya memperpanjang siklus pengujian dan menambah beban kerja tim.

3.4 Solusi atas Kendala yang Ditemukan

3.4.1 Meningkatkan Pemahaman terhadap Prioritas Bug

Untuk mengatasi ketidakpahaman terhadap prioritas bug, langkah-langkah berikut dapat diambil:

1. Pelatihan Internal

Mengadakan sesi pelatihan internal yang menjelaskan sistem prioritas bug, termasuk bagaimana menilai dampak bug terhadap fungsionalitas sistem. QA baru dapat dilibatkan dalam simulasi kasus untuk memahami bagaimana menentukan prioritas bug dengan tepat.

2. Dokumentasi Panduan

Membuat panduan internal yang terstruktur mengenai prioritas bug, termasuk definisi setiap kategori (*High, Medium, Low*) beserta contohnya, sehingga QA memiliki referensi yang jelas saat menganalisis bug.

3. Pendampingan QA Senior dan developer

Memberikan kesempatan bagi QA baru untuk bekerja secara langsung dengan QA senior dalam mengidentifikasi dan menentukan prioritas bug, sehingga mereka dapat belajar dari pengalaman nyata. Kemudian berdasarkan pemaparan eksekutif FOS, tim developer ialah yang paling mengerti prioritas dari suatu fitur software. Sehingga keharusan developer untuk melabeli suatu bug dengan cepat dan tepat sangat dibutuhkan.

3.4.2 Mengelola Perubahan Mayor pada Hashigake

Menghadapi perubahan mayor pada aplikasi selama proses otomatisasi memerlukan pendekatan yang terorganisir untuk meminimalkan gangguan dan mempercepat adaptasi. Solusi berikut dapat diterapkan:

1. Version Control pada Skrip Otomasi

Mengelola skrip otomatisasi dengan menggunakan sistem *version control* seperti Git untuk melacak perubahan, memungkinkan rollback jika skrip lama diperlukan.

2. Komunikasi Proaktif dengan Tim Pengembang

QA perlu memiliki komunikasi yang terintegrasi dengan tim pengembang untuk mendapatkan informasi awal tentang perubahan besar yang akan dilakukan, sehingga mereka dapat mempersiapkan pengujian yang relevan sebelum perubahan diterapkan.

3. Penggunaan Framework yang Fleksibel

Memfaatkan framework pengujian yang mendukung *parameterization* dan *modularization* untuk mempermudah penyesuaian skrip otomatisasi terhadap perubahan fitur atau elemen UI. Sehingga ketika ada perubahan pada test document, tim QA yang menangani otomatisasi dapat langsung mengetahui parameter apa yang perlu diperbarui baik di dalam code maupun di name range automation test document.

4. Buffer Time untuk Testing

Untuk menangani perubahan mayor pada aplikasi Hashigake, tim automation QA menyediakan waktu khusus dalam jadwal proyek sebagai buffer time. Waktu ini digunakan untuk mencoba kembali seluruh elemen pada website guna memastikan stabilitas setelah perubahan diterapkan. Dalam praktiknya, tim automation QA menghabiskan sekitar satu hari untuk menguji ulang elemen-elemen website, memeriksa apakah perubahan fitur juga memengaruhi logika yang mendasarinya. Fokus utama adalah pada fitur-fitur yang mengalami perubahan, untuk memastikan bahwa logika baru berjalan sesuai dengan ekspektasi tanpa mengganggu fungsi lain. Pendekatan ini membantu meminimalkan risiko bug tersembunyi yang mungkin muncul akibat perubahan besar pada sistem.