

**SISTEM PENCEGAH PLAGIARISME DALAM TUGAS MAHASISWA
DENGAN SNAPSHOTTING DAN USER ACTIVITY LOGGING**

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
AZZAM SYAWQI AZIZ
NIM: 155150207111132



PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

SISTEM PENCEGAH PLAGIARISME DALAM TUGAS MAHASISWA DENGAN
SNAPSHOTTING DAN USER ACTIVITY LOGGING

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Azzam Syawqi Aziz
NIM : 155150207111132

Skripsi ini telah diuji dan dinyatakan lulus pada:

2 Juli 2019

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I



Fajar Pradana, S.ST, M.Eng
NIP: 19871121 201504 1 004

Dosen Pembimbing II



Dr.Eng. Fitra Abdurrachman Bachtiar, S.T, M.Eng
NIP: 198406282019031006

Mengetahui

Ketua Jurusan Teknik Informatika



Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Juli 2019



Azzam Syawqi Aziz

NIM : 155150207111132

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga skripsi yang berjudul “SISTEM PENCEGAH PLAGIARISME DALAM TUGAS MAHASISWA DENGAN SNAPSHOTTING DAN USER ACTIVITY LOGGING” ini dapat terselesaikan. Dengan selesainya penulisan skripsi ini, penulis ingin menyampaikan terima kasih kepada:

1. Allah SWT yang telah memberikan nikmat kesehatan, kekuatan dan kemudahan sehingga penulis dapat menyelesaikan skripsi ini.
2. Ustaz, orang tua dan seluruh keluarga besar penulis yang senantiasa mendukung dan mendukung demi terselesaikannya skripsi ini.
3. Bapak Fajar Pradana, S.ST, M.Eng, selaku Dosen Pembimbing I yang telah me luangkan waktu untuk membimbing dengan semangat dan sabar sehingga penulis dapat menyelesaikan skripsi ini.
4. Bapak Dr.Eng. Fitra Abdurrachman Bachtiar, S.T, M.Eng, selaku Dosen Pembimbing II yang telah mengarahkan dan membimbing penulis dengan dengan semangat dan sabar sehingga penulis dapat menyelesaikan skripsi ini.
5. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D dan Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Jurusan Teknik Informatika dan Kepala Program Studi Teknik Informatika Fakultas Ilmu Komputer, Universitas Brawijaya.
6. Seluruh pengembang kakas bantu perangkat lunak bebas yang digunakan dalam skripsi ini.
7. Seluruh civitas akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penyelesaian skripsi ini.

Penulis menyadari bahwa dalam penelitian ini masih banyak kekurangan. Oleh karena itu, saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap penelitian ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 2 Juli 2019

Penulis
azzamsa@student.ub.ac.id

ABSTRAK

Azzam Syawqi Aziz, Sistem Pencegah Plagiarisme Dalam Tugas Mahasiswa dengan snapshotting dan User Activity Logging

Pembimbing: Fajar Pradana, S.ST, M.Eng dan Dr.Eng. Fitra Abdurrachman Bachtiar, S.T, M.Eng

Plagiarisme merupakan hal yang tak terpisahkan dari lingkungan akademisi. Beragam motivasi termasuk bekerja paruh waktu dan dilema sosial menyumbang tingginya persentase plagiarisme di tingkat universitas. Plagiarisme juga telah menumbangkan karier tokoh-tokoh besar. Penanganan plagiarisme dengan metode *plagiarism detection* memiliki banyak kekurangan seperti timbulnya paradigma polisi-kriminal antara murid dan guru, pola pikir nilai *oriented*, dan beberapa kelemahan teknis lain seperti lemah dalam menganalisis dokumen yang telah diterjemahkan. Hellas, Leinonen, dan Ihantola telah melakukan penelitian plagiarisme dengan metode *plagiarism prevention*, tetapi metode yang dilakukan memiliki beberapa kekuatan, yaitu mengabaikan proses penggerjaan tugas sehingga dapat dicurangi siswa dengan membeli tugas, analisis hanya dilakukan di akhir sehingga siswa dapat berkolaborasi melakukan kecurangan selama tugas dikerjakan, tidak memiliki *log* sehingga guru tidak dapat menilai usaha siswa, dan perangkat lunak yang digunakan terikat dengan *platform* tertentu. Penelitian ini bertujuan untuk menyempurnakan penelitian yang telah dilakukan oleh Hellas, Leinonen, dan Ihantola dengan menambahkan proses *snapshotting* yang merekam seluruh perubahan berkas tugas, dan *user activity logging* yang merekam seluruh aktivitas siswa selama mengerjakan tugas. Sistem yang dibangun juga bersifat *agnostic* sehingga guru memiliki kebebasan memilih bentuk tugas dan siswa bebas menggunakan perangkat lunak disukai untuk mengerjakan tugas. Pengujian sistem dilakukan dengan pengujian unit, integrasi, validasi, *automated testing*, dan pengujian *compatibility*. Pengujian unit dan integrasi bernilai valid dan mencapai nilai di atas 70% *code coverage*, pengujian validasi pada 34 kasus uji menghasilkan hasil valid, dan pengujian *compatibility* menunjukkan sistem dapat berjalan pada enam *desktop environment* umum. Hasil *automated testing* yang valid dan tidak memiliki galat dapat mempercepat proses pengujian.

Kata kunci: *plagiarism prevention, snapshotting, user activity logging*

ABSTRACT

Azzam Syawqi Aziz, Sistem Pencegah Plagiarisme Dalam Tugas Mahasiswa dengan snapshotting dan User Activity Logging

Supervisors: Fajar Pradana, S.ST, M.Eng and Dr.Eng. Fitra Abdurrachman Bachtiar, S.T, M.Eng

Plagiarism is an inseparable thing from an academic environment. Various motivations including working part-time and social dilemmas contribute to the high percentage of plagiarism at the university level. Plagiarism has also toppled the careers of great figures. Handling plagiarism by plagiarism detection the method has many disadvantages such as the emergence of the police-criminal paradigm between students and teachers, value-oriented mindset, and some other technical weaknesses such as inability in analyzing documents that have been translated. Hellas, Leinonen, and Ihantola have done plagiarism research with the plagiarism prevention method, but the method that is carried out has several disadvantages, namely ignoring the work process so that students can be cheated by buying assignments, the analysis is only done at the end so students can collaborate cheating during assignments, no any log produced so the teacher can't assess student effort, and the software used depends on a certain platform. This research aims to perfect the research done by Hellas, Leinonen, and Ihantola by adding the snapshotting process which records all document changes, and user activity logging which records all student activities while working on a task. The system is also agnostic so that the teacher has the freedom to choose the form of assignments and students are free to use the preferred software to work on the task. The system is tested using unit testing, integration, validation, automated testing, and compatibility testing. Unit testing and integration are valid and scored above 70% code coverage, validation testing in 34 test cases produced valid results, and the compatibility test shows that the system able to run on six general desktop environment. The automated testing result is valid and does not contain errors can speed up the testing process.

Keywords: *plagiarism prevention, snapshotting, user activity logging*

DAFTAR ISI

PENGESAHAN	ii
PERNYATAAN ORISINALITAS	iii
KATA PENGANTAR	iv
ABSTRAK	v
ABSTRACT	vi
DAFTAR ISI	x
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xv
DAFTAR LAMPIRAN	xv
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	4
1.3 Tujuan	4
1.4 Manfaat	4
1.5 Batasan Masalah	5
1.6 Sistematika Pembahasan	5
BAB 2 LANDASAN KEPUSTAKAAN	7
2.1 Kajian Pustaka	7
2.2 Rekayasa Perangkat Lunak	8
2.3 Pengembangan Perangkat Lunak	9
2.3.1 Model <i>Waterfall</i>	9
2.3.2 Arsitektur <i>Model View Controller</i>	11
2.3.3 Pendekatan Berorientasi Objek	11
2.3.4 Pemodelan Berorientasi Objek	12
2.3.4.1 <i>Use Case Diagram</i>	13
2.3.4.2 <i>Sequence Diagram</i>	13
2.3.4.3 <i>Class Diagram</i>	15
2.3.5 Pengujian Perangkat Lunak	17

2.3.5.1	Tahapan Pengujian	17
2.3.5.2	Metode Pengujian	18
2.3.5.3	Teknik Pengujian	19
2.3.5.4	<i>Automated Testing</i>	24
2.3.5.5	Pengujian <i>Compatibility</i>	25
2.4	<i>Snapshotting</i>	25
2.5	<i>User Activity Logging</i>	26
2.6	<i>Edit-distance Algorithm</i>	27
2.7	Teknologi Pengembangan Perangkat Lunak	28
2.7.1	<i>Python</i>	28
2.7.2	<i>Git (Distributed Version Control)</i>	28
2.7.3	<i>Qt (Widget Toolkit)</i>	29
2.7.4	<i>Draw.io</i>	30
BAB 3 METODOLOGI		31
3.1	Studi Literatur	31
3.2	Rekayasa Kebutuhan	32
3.3	Perancangan	33
3.4	Implementasi	34
3.5	Pengujian	35
3.6	Penarikan Kesimpulan	36
BAB 4 REKAYASA KEBUTUHAN		37
4.1	Deskripsi Umum Sistem	37
4.2	Analisis Kebutuhan	38
4.3	Identifikasi Aktor	38
4.4	Kebutuhan Fungsional Sistem	39
4.5	Kebutuhan Non-Fungsional Sistem	52
4.6	Pemodelan Kebutuhan	52
4.6.1	<i>Use Case Diagram</i>	52
4.6.2	<i>Use Case Scenario</i>	54
BAB 5 PERANCANGAN DAN IMPLEMENTASI		63
5.1	Perancangan Sistem	63
5.1.1	Perancangan Arsitektur	63
5.1.1.1	<i>Sequence Diagram</i> Merekam Penggeraan Tugas . .	63
5.1.1.2	<i>Sequence Diagram</i> Melihat Hasil Rekaman	64

5.1.1.3	<i>Sequence Diagram</i> Melihat Grafik <i>Edit-Distance</i> Tugas Sebelumnya	64
5.1.1.4	<i>Class Diagram</i>	64
5.1.2	Perancangan Komponen	64
5.1.2.1	Perancangan Komponen <i>Class Controller</i>	70
5.1.2.2	Perancangan Komponen <i>Class LogController</i>	70
5.1.2.3	Perancangan Komponen <i>Class SearchController</i>	71
5.1.3	Perancangan Antarmuka	72
5.1.3.1	Perancangan Antarmuka Tampilan <i>Tray</i> Pada Sistem <i>Lup Recorder</i>	72
5.1.3.2	Perancangan Antarmuka Tampilan Hasil Rekaman Pada Sistem <i>Lup Viewer</i>	73
5.1.3.3	Perancangan Antarmuka Tampilan <i>Edit-distance</i> Tugas Sebelumnya Pada Sistem <i>Lup Viewer</i>	75
5.2	Implementasi Sistem	76
5.2.1	Spesifikasi Sistem	76
5.2.2	Implementasi Kode Program	77
5.2.2.1	Implementasi Kode Program <i>Class Controller</i> Pada Sistem <i>Lup Recorder</i>	77
5.2.2.2	Implementasi Kode Program <i>Class LogController</i> Pada Sistem <i>Lup Viewer</i>	78
5.2.2.3	Implementasi Kode Program <i>Class SearchController</i> Pada Sistem <i>Lup Viewer</i>	79
5.2.3	Implementasi Antarmuka	79
5.2.3.1	Implementasi Antarmuka Tampilan <i>Tray</i> Pada Sistem <i>Lup Recorder</i>	80
5.2.3.2	Implementasi Antarmuka Tampilan Hasil Rekaman Pada Sistem <i>Lup Viewer</i>	80
5.2.3.3	Perancangan Antarmuka Tampilan <i>Edit-distance</i> Tugas Sebelumnya Pada Sistem <i>Lup Viewer</i>	80
BAB 6 PENGUJIAN		82
6.1	Pengujian Unit	82
6.1.1	Pengujian Unit <i>Class Controller</i> Function <i>get_all_windows</i>	82
6.1.2	Pengujian Unit <i>Class LogController</i> Function <i>populate_logs</i>	85
6.1.3	Pengujian Unit <i>Class SearchController</i> Function <i>construct_ed_graph_path</i>	88
6.2	Pengujian Integrasi	90

6.3	Pengujian Validasi	93
6.3.1	Pengujian Validasi Merekam Pengerjaan Tugas	93
6.3.2	Pengujian Validasi Mengubah Interval Rekaman	94
6.3.3	Pengujian Validasi Melihat Seluruh Daftar Rekaman	96
6.3.4	Pengujian Validasi Melihat Hasil Rekaman	98
6.3.5	Pengujian Validasi Mencari Tersangka	100
6.3.6	Pengujian Validasi Mencari <i>Window</i>	103
6.3.7	Pengujian Validasi Melihat Seluruh Alamat <i>IP</i>	104
6.3.8	Pengujian Validasi Melihat Grafik <i>Edit-distance</i>	105
6.3.9	Pengujian Validasi Melihat Grafik <i>Edit-distance</i> Tugas Sebelumnya	106
6.3.10	Pengujian Validasi Menyimpan Grafik <i>Edit-distance</i>	108
6.3.11	Pengujian Validasi Mengekskor Semua Nilai <i>Edit-distance</i>	109
6.3.12	Pengujian Validasi Mengalihkan Format Waktu	110
6.4	<i>Automated Testing</i>	111
6.4.1	<i>Test Script Automated Testing Class Controller Function get_all_windows</i>	111
6.4.2	<i>Test Script Automated Testing Class LogController Function populate_logs</i>	112
6.4.3	<i>Test Script Automated Testing Class SearchController Function construct_ed_graph_path</i>	113
6.5	Pengujian <i>Compatibility</i>	114
	BAB 7 PENUTUP	118
7.1	Kesimpulan	118
7.2	Saran	119

DAFTAR TABEL

Tabel 2.1	Notasi <i>use case diagram</i> (Rumbaugh, Jacobson, & Booch, 2004)	13
Tabel 2.2	Notasi <i>sequence diagram</i>	14
Tabel 2.3	Notasi <i>class diagram</i>	16
Tabel 2.4	<i>Test case</i> dari contoh <i>pseudocode</i> 2.1	22
Tabel 4.5	Identifikasi aktor pada sistem <i>Lup Viewer</i>	39
Tabel 4.6	Identifikasi aktor pada sistem <i>Lup Recorder</i>	39
Tabel 4.7	Spesifikasi kebutuhan fungsional, definisi, dan pemetaan nama <i>use case</i> sistem <i>Lup Recorder</i>	40
Tabel 4.8	Spesifikasi kebutuhan fungsional, definisi, dan pemetaan nama <i>use case</i> sistem <i>Lup Viewer</i>	42
Tabel 4.9	Spesifikasi kebutuhan non-fungsional, parameter, dan deskripsi kebutuhan sistem <i>Lup Recorder</i>	52
Tabel 4.10	Spesifikasi kebutuhan non-fungsional, parameter, dan deskripsi kebutuhan sistem <i>Lup Viewer</i>	52
Tabel 4.11	<i>Use case scenario</i> untuk Merekam Pengerajan Tugas	54
Tabel 4.12	<i>Use case scenario</i> untuk Mengubah Interval Rekaman	55
Tabel 4.13	<i>Use case scenario</i> untuk Melihat Seluruh Daftar Rekaman	55
Tabel 4.14	<i>Use case scenario</i> untuk Menampilkan Hasil Rekaman	56
Tabel 4.15	<i>Use case scenario</i> untuk Mencari Tersangka.	57
Tabel 4.16	<i>Use case scenario</i> untuk Mencari <i>Window</i>	58
Tabel 4.17	<i>Use case scenario</i> untuk Menampilkan Seluruh Alamat IP	59
Tabel 4.18	<i>Use case scenario</i> untuk Melihat Grafik <i>Edit-distance</i>	59
Tabel 4.19	<i>Use case scenario</i> untuk Melihat Grafik <i>Edit-distance</i> Tugas Sebelumnya	60
Tabel 4.20	<i>Use case scenario</i> untuk Menyimpan Grafik <i>Edit-distance</i>	61
Tabel 4.21	<i>Use case scenario</i> untuk Mengekspor Semua Nilai <i>Edit-distance</i>	61
Tabel 4.22	<i>Use case scenario</i> untuk Mengalihkan Format Waktu	62
Tabel 5.23	Perancangan antarmuka tampilan <i>Tray</i> sistem <i>Lup Recorder</i>	72
Tabel 5.24	Penjelasan antarmuka tampilan Hasil Rekaman sistem <i>Lup Viewer</i>	73
Tabel 5.25	Penjelasan antarmuka tampilan <i>Edit-distance</i> Tugas Sebelumnya	75
Tabel 5.26	Spesifikasi perangkat keras	76
Tabel 5.27	Spesifikasi perangkat lunak	77

Tabel 6.28	Kasus uji dan hasil uji <i>function get_all_windows</i>	84
Tabel 6.29	Kasus uji dan hasil uji <i>function populate_logs</i>	87
Tabel 6.30	Kasus uji dan hasil uji <i>function construct_ed_graph_path</i> . .	89
Tabel 6.31	Kasus uji dan hasil uji <i>function populate_logs</i>	92
Tabel 6.32	Kasus uji dan hasil uji Merekam Pengerajan Tugas	93
Tabel 6.33	Kasus uji dan hasil uji Merekam Pengerajan Tugas alternatif 1	93
Tabel 6.34	Kasus uji dan hasil uji Merekam Pengerajan Tugas alternatif 2	94
Tabel 6.35	Kasus uji dan hasil uji Mengubah Interval Rekaman (<i>valid input equivalence partitioning</i>)	95
Tabel 6.36	Kasus uji dan hasil uji Mengubah Interval Rekaman alternatif 1 (<i>invalid input equivalence partitioning</i>)	95
Tabel 6.37	Kasus uji dan hasil uji Mengubah Interval Rekaman (<i>valid input boundary value analysis</i>)	96
Tabel 6.38	Kasus uji dan hasil uji Mengubah Interval Rekaman alternatif 1 (<i>invalid input boundary value analysis</i>)	96
Tabel 6.39	Kasus uji dan hasil uji Melihat Seluruh Daftar Rekaman . . .	97
Tabel 6.40	Kasus uji dan hasil uji Melihat Seluruh Daftar Rekaman alternatif 1	97
Tabel 6.41	Kasus uji dan hasil uji Melihat Seluruh Daftar Rekaman alternatif 2	97
Tabel 6.42	Kasus uji dan hasil uji Melihat Hasil Rekaman	98
Tabel 6.43	Kasus uji dan hasil uji Melihat Hasil Rekaman alternatif 1 . .	99
Tabel 6.44	Kasus uji dan hasil uji Melihat Hasil Rekaman alternatif 2 . .	99
Tabel 6.45	Kasus uji dan hasil uji Melihat Hasil Rekaman alternatif 3 . .	100
Tabel 6.46	Kasus uji dan hasil uji Mencari Tersangka (<i>valid input equivalence partitioning</i>)	101
Tabel 6.47	Kasus uji dan hasil uji Mencari Tersangka alternatif 1 (<i>invalid input equivalence partitioning</i>)	101
Tabel 6.48	Kasus uji dan hasil uji Mencari Tersangka (<i>valid input boundary value analysis</i>)	102
Tabel 6.49	Kasus uji dan hasil uji Mencari Tersangka alternatif 1 (<i>invalid input boundary value analysis</i>)	102
Tabel 6.50	Kasus uji dan hasil uji Mencari Tersangka alternatif 2	102
Tabel 6.51	Kasus uji dan hasil uji Mencari Tersangka alternatif 3	103
Tabel 6.52	Kasus uji dan hasil uji Mencari <i>Window</i>	103
Tabel 6.53	Kasus uji dan hasil uji Mencari <i>Window</i> alternatif 1	104
Tabel 6.54	Kasus uji dan hasil uji Mencari <i>Window</i> alternatif 2	104

Tabel 6.55	Kasus uji dan hasil uji Melihat Seluruh Alamat <i>IP</i>	105
Tabel 6.56	Kasus uji dan hasil uji Melihat Grafik <i>Edit-distance</i>	105
Tabel 6.57	Kasus uji dan hasil uji Melihat Grafik <i>Edit-distance</i> alternatif	1106
Tabel 6.58	Kasus uji dan hasil uji Melihat Grafik <i>Edit-distance</i> Tugas Se- belumnya	106
Tabel 6.59	Kasus uji dan hasil uji Melihat Grafik <i>Edit-distance</i> Tugas Se- belumnya alternatif 1	107
Tabel 6.60	Kasus uji dan hasil uji Melihat Grafik <i>Edit-distance</i> Tugas Se- belumnya alternatif 2	107
Tabel 6.61	Kasus uji dan hasil uji Melihat Grafik <i>Edit-distance</i> Tugas Se- belumnya alternatif 3	108
Tabel 6.62	Kasus uji dan hasil uji Menyimpan Grafik <i>Edit-distance</i> . . .	109
Tabel 6.63	Kasus uji dan hasil uji Mengekspor Semua Nilai <i>Edit-distance</i>	109
Tabel 6.64	Kasus uji dan hasil uji Mengekspor Semua Nilai <i>Edit-distance</i> alternatif 1	110
Tabel 6.65	Kasus uji dan hasil uji Mengalihkan Format Waktu	110
Tabel 6.66	Daftar <i>desktop environment</i> yang diuji	114

DAFTAR GAMBAR

Gambar 2.1	Model <i>waterfall</i> (Sommerville, 2014)	11
Gambar 2.2	Contoh <i>use case diagram</i> (Pressman, 2010)	14
Gambar 2.3	Contoh <i>sequence diagram</i> (Pressman, 2010)	15
Gambar 2.4	Contoh <i>class diagram</i> (Pressman, 2010)	16
Gambar 2.5	Tahapan pengujian (Presman, 2010)	18
Gambar 2.6	Notasi <i>flow graph</i> (Presman, 2010)	21
Gambar 2.7	Representasi <i>control flow graph</i> dari <i>pseudocode</i> 2.1	22
Gambar 2.8	Contoh <i>equivalence partitioning</i> (Sommerville, 2014)	23
Gambar 2.9	Contoh <i>boundary value analysis</i> (Sommerville, 2014)	24
Gambar 2.10	Penyimpanan data sebagai <i>snapshot</i> dari waktu ke waktu (Chacon & Straub, 2014)	26
Gambar 2.11	Grafik <i>edit-distance</i>	28
Gambar 2.12	<i>Window</i> sederhana dengan <i>Qt</i>	30
Gambar 3.13	Diagram alir metodologi	31
Gambar 4.14	Gambaran umum sistem	37
Gambar 4.15	<i>Use case diagram Lup Recorder</i>	53
Gambar 4.16	<i>Use case diagram Lup Viewer</i>	53
Gambar 5.17	<i>Sequence diagram</i> Merekam Pengerjaan Tugas	65
Gambar 5.18	<i>Sequence diagram</i> Melihat Hasil Rekaman	66
Gambar 5.19	<i>Sequence diagram</i> Melihat Grafik <i>Edit-Distance</i> Tugas Sebelumnya	67
Gambar 5.20	Pemodelan <i>class diagram</i> <i>Lup Recorder</i>	68
Gambar 5.21	Pemodelan <i>class diagram</i> <i>Lup Viewer</i>	69
Gambar 5.22	Perancangan antarmuka tampilan <i>Tray</i> sistem <i>Lup Recorder</i>	72
Gambar 5.23	Perancangan antarmuka tampilan Hasil Rekaman sistem <i>Lup Viewer</i>	73
Gambar 5.24	Perancangan antarmuka tampilan <i>Edit-distance</i> Tugas Sebelumnya	75
Gambar 5.25	Implementasi antarmuka tampilan <i>Tray</i> pada sistem <i>Lup Recorder</i>	80
Gambar 5.26	Implementasi antarmuka tampilan Hasil Rekaman pada sistem <i>Lup Viewer</i>	81
Gambar 5.27	Perancangan antarmuka tampilan <i>Edit-distance</i> Tugas Sebelumnya pada sistem <i>Lup Viewer</i>	81
Gambar 6.28	Pengujian unit mencapai 100% <i>code coverage</i>	82

Gambar 6.29	<i>Flow graph</i> dari <i>pseudocode get_all_windows</i>	83
Gambar 6.30	<i>Flow graph</i> dari <i>pseudocode pupoulate_logs</i>	86
Gambar 6.31	<i>Flow graph</i> dari <i>pseudocode construct_ed_graph_path</i>	88
Gambar 6.32	<i>Flow graph</i> dari <i>pseudocode pupoulate_logs</i>	90
Gambar 6.33	Seluruh hasil <i>automated testing</i>	111
Gambar 6.34	Sistem <i>Lup Recorder</i> berhasil merekam berkas tugas pada <i>desktop environment</i> yang berbeda-beda	116
Gambar 6.35	<i>Lup Viewer</i> pada <i>desktop environment i3wm</i>	117
Gambar 6.36	<i>Lup Viewer</i> pada <i>desktop environment XFCE</i>	117

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Plagiarisme dalam lingkungan akademisi adalah penggunaan kata atau ide dari sebuah sumber tanpa menyertakan pengakuan sebagaimana ditentukan oleh prinsip-prinsip akademik (Meuschke & Gipp, 2013). Meuschke & Gipp (2013) berpendapat bahwa plagiarisme dalam lingkungan akademisi tidak selalu dapat dikaitkan dengan pencurian. Plagiarisme tersebut dapat terjadi karena unsur tidak sengaja, seperti penulis yang lupa memasukkan sitiran, melakukan sitiran pada sumber yang salah, atau tidak sengaja melakukan sitiran pada dirinya sendiri. Terdapat beberapa tipe plagiarisme, diantaranya menyalin kata, parafrase, dan menerjemahkan kata (Kiss, 2013).

Plagiarisme menjadi hal yang seakan tak terpisahkan di lingkungan akademisi. Mahasiswa seringkali melakukan plagiarisme dalam tugas maupun ujian. Ketersediaan sumber daya *internet* dan kemudahan mencari informasi berkontribusi pada munculnya plagiarisme oleh siswa (Born, 2003). Plagiarisme di Indonesia tidak hanya dilakukan oleh mahasiswa saja, melainkan juga peneliti dan dosen dari universitas-universitas ternama (Agustina & Raharjo, 2017). Kiss (2013) juga menyampaikan bahwa banyak insiden plagiarisme yang menyebabkan karier tokoh-tokoh besar hancur. Banyaknya motivasi untuk melakukan plagiarisme di tingkat universitas, seperti mahasiswa yang bekerja paruh waktu dan dilema sosial, membuat tingkat plagiarisme di universitas semakin tinggi (Park, 2004). Oleh karena alasan-alasan tersebut diperlukan penanganan plagiarisme.

Metode dalam plagiarisme dapat dikategorikan menjadi dua. Pertama adalah metode pendektsian atau *plagiarism detection*. Metode ini menitikberatkan pada proses pendektsian apakah seorang siswa melakukan plagiarisme setelah siswa tersebut menyelesaikan aktivitas yang memungkinkan terjadinya plagiarisme, seperti kuis tugas rumah (*take home quiz*), ujian harian dan ujian akhir. Hasil dari tugas atau ujian tersebut akan melewati proses pendektsian plagiarisme, seperti menggunakan alat bantu perangkat lunak *plagiarism detection software* (Neill & Shanmuganthan, 2004) atau dilakukan secara manual. Metode kedua adalah metode pencegahan atau *plagiarism prevention*. Penanganan plagiarisme pada metode pencegahan ditekankan pada saat sebelum siswa memulai aktivitas atau selama mengerjakan aktivitas yang memungkinkan terjadinya plagiarisme. Proses pencegahannya, seperti mengimplementasikan *formative assessment* pada siswa (Leung & Cheng, 2017), mengedukasi siswa tentang plagiarisme, mengajarkan cara sitir-

an dengan benar, membangun atmosfer yang sehat di kelas (McLafferty & Foust, 2004), dan melakukan pencegahan dengan bantuan perangkat lunak (Hellas, Leinonen, & Ihantola, 2017).

Penggunaan *Plagiarism Detection Software* atau *PDS* memudahkan para guru untuk menemukan siswa tersangka plagiarisme. Tetapi penggunaan *PDS* memiliki kekurangan, seperti dapat menyebabkan paradigma siswa kepada guru bagaikan hubungan polisi-kriminal daripada guru-murid (Howard, 2002), membentuk pola pikir siswa menjadi nilai *oriented* dari pada *goal oriented* (Dweck & Leggett, 1988), membuat siswa mencari cara-cara baru agar tidak terdeteksi *PDS* (Kiss, 2013); mengubah teks menjadi gambar, membubuhkan *invinsible character* dan mengubah *character map*, yaitu mengubah huruf "a" menjadi karakter "c" dan mengubah huruf lain ke karakter lain; lemah dalam menganalisis dokumen yang sudah diterjemahkan (Meuschke & Gipp, 2013), memiliki batasan dalam proses menganalisis berbagai macam gaya sitiran (Modiba, Pieterse, & Haskins, 2016) dan berbagai macam bahasa (Martins, et al., 2014), tidak dapat melacak bantuan eksternal yang datang dari luar kelas (Hellas, Leinonen, & Ihantola, 2017), harga yang ditawarkan tidak terjangkau sehingga negara berkembang tidak dapat mengadopsinya (Agustina & Raharjo, 2017; Modiba, Pieterse, & Haskins, 2016) dan pelanggaran hak-hak intelektual dan hak milik siswa seperti yang dilakukan *Turnitin*. *Turnitin* menyalin dan menyimpan seluruh tugas siswa ke dalam basis datanya, terlepas siswa tersebut melakukan plagiarisme atau tidak. Oleh karena itu beberapa universitas meninggalkannya (Park, 2004).

PDS tidak dapat mendeteksi semua kasus yang ada pada permasalahan plagiarisme (Martins, et al., 2014). Oleh karena itu, pencegahan plagiarisme sangat diutamakan (Garner, et al., 2012). Metode pencegahan plagiarisme (*plagiarism prevention*) dengan bantuan perangkat lunak memberikan kemampuan pada guru untuk menganalisis kecurangan selama proses penggerjaan tugas, seperti mendeteksi plagiarisme dari sumber non-digital dan bantuan eksternal. Selain itu, guru dapat menganalisis pola proses penggerjaan dari *log* yang dihasilkan sehingga dapat menganalisis adanya kejanggalan maupun menemukan siswa yang lemah pada pelajaran tertentu. Kemampuan-kemampuan tersebut tidak dapat dilakukan oleh *PDS* (Hellas, Leinonen, & Ihantola, 2017).

Hellas, Leinonen, & Ihantola (2017) menggunakan metode pencegahan plagiarisme (*plagiarism prevention*) dengan bantuan perangkat lunak. Metode tersebut merekam waktu mulai, waktu selesai dan alamat *IP* siswa dengan bantuan perangkat lunak *JPlag*. Pada akhir prosesnya dilakukan analisis hasil tugas menggunakan algoritme *edit-distance*. Rekaman waktu mulai dan waktu selesai digu-

nakan untuk menemukan siswa yang mengerjakan pada waktu yang bersamaan, rekaman alamat *IP* digunakan untuk menenemukan siswa yang mengerjakan pada tempat yang sama, dan analisis hasil akhir tugas menggunakan *edit-distance* digunakan untuk menemukan siswa yang mendapatkan bantuan dari sumber eksternal atau tidak dikerjakan dengan kemampuannya sendiri. Tetapi metode ini memiliki beberapa kelemahan, seperti tidak merekam segala aktivitas yang dilakukan selama mengerjakan tugas sehingga guru hanya menganalisis hasil akhir tugas dan mengabaikan proses pengerjaan tugas. Hal ini dapat dicurangi siswa dengan menyelesaikan tugas dengan membeli (Leung & Cheng, 2017). Analisis bantuan eksternal hanya dilakukan pada hasil akhir tugas sehingga kecurangan selama mengerjakan tugas dapat dicurangi dengan bantuan mencari jawaban melalui *browser* maupun berkolaborasi menggunakan sosial media. Metode ini juga tidak memiliki *log* pengerjaan sehingga guru tidak dapat menilai usaha yang telah dilakukan siswa (Hellas, Leinonen, & Ihantola, 2017). Selain itu, proses rekaman dilakukan dengan bantuan perangkat lunak yang terikat dengan *platform* tertentu sehingga memiliki batasan bentuk tugas yang akan diberikan dan batasan pilihan kakas bantu yang akan digunakan siswa.

Kelemahan yang ada pada metode yang dilakukan Hellas, Leinonen, & Ihantola (2017) akan disempurnakan dengan menyelesaikan masalah yang telah disebutkan sebelumnya dengan cara menambahkan proses *screenshotting* dan *user activity logging* selama pengerjaan tugas. Proses *screenshotting* akan merekam seluruh perubahan dokumen. Oleh karena itu, *screenshotting* dapat menghindari adanya siswa yang melakukan kecurangan melalui bantuan eksternal seperti membeli tugas dari orang lain. *User activity logging* merekam seluruh aktivitas siswa selama pengerjaan tugas, seperti nama *login*, nama peranti yang digunakan, *window* yang sedang aktif, dan seluruh *window* yang terbuka. Maka *user activity logging* dapat menghindari adanya bantuan eksternal melalui *browser* maupun perangkat lunak sosial media ketika proses mengerjakan tugas. Proses perekaman data menggunakan perangkat lunak yang bersifat *agnostic* sehingga guru tidak memiliki batasan dalam memilih bentuk tugas yang akan diberikan dan siswa memiliki kebebasan menggunakan kakas bantu yang disukai. Selain itu, seluruh proses pengerjaan tugas direkam sehingga guru dapat menganalisis seberapa besar kesuguhan dan seberapa banyak keterlibatan siswa dalam tugas yang diberikan.

Penelitian ini akan mengimplementasikan metode *screenshotting* dan *user activity logging*, dengan tambahan algoritme *edit-distance* yang didapatkan dari penelitian sebelumnya. Metode *screenshotting* akan dibangun di atas *distributed version control* untuk merekam seluruh perubahan dokumen tugas. Metode *acti-*

activity logging digunakan untuk merekam segala aktivitas siswa selama penggerjaan tugas. Algoritme *edit-distance* digunakan untuk menghitung nilai *edit-distance* pada berkas tugas. Maka seluruh usaha, keputusan, keterlibatan dan aktivitas siswa dapat dianalisis oleh guru.

1.2 Rumusan Masalah

Berdasarkan permasalahan yang diangkat, maka dapat dirumuskan masalah sebagai berikut:

1. Bagaimana hasil analisis kebutuhan dalam pengembangan aplikasi pencegah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging*?
2. Bagaimana hasil perancangan dan implementasi aplikasi pencegah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging*?
3. Bagaimana hasil pengujian dari pengembangan aplikasi pencegah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging*?

1.3 Tujuan

Tujuan penelitian ini adalah sebagai berikut:

1. Menganalisis kebutuhan yang diperlukan untuk membangun aplikasi pencegah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging*.
2. Merancang dan mengimplementasikan aplikasi pencegah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging* sesuai dengan hasil analisis kebutuhan.
3. Menguji aplikasi pencegah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging* untuk memastikan bahwa aplikasi yang dibangun sesuai dengan hasil rancangan dan kebutuhan sebelumnya.

1.4 Manfaat

Manfaat yang didapatkan dari penelitian ini adalah:

1. Menyediakan aplikasi yang dapat mencegah siswa melakukan plagiarisme.

2. Menyediakan aplikasi yang dapat memberikan petunjuk siswa yang melakukan kecurangan selama mengerjakan tugas.
3. Menyediakan aplikasi yang dapat memberikan petunjuk siswa yang lemah pada suatu pelajaran tertentu.

1.5 Batasan Masalah

Penelitian ini memiliki batasan-batasan sebagai berikut:

1. Sistem yang dikembangkan hanya dapat mengolah tugas dengan bentuk *plain text format*, sistem tidak dapat mengolah *binary format*.
2. Sistem yang dikembangkan hanya dapat berjalan pada sistem operasi *GNU-Linux*.
3. Sistem yang dikembangkan membutuhkan perangkat lunak pengolah kata yang memiliki fitur *autosave*.

1.6 Sistematika Pembahasan

Penjelasan singkat mengenai pembahasan dan langkah-langkah yang dilakukan dalam setiap bab pada penelitian ini tersusun sebagai berikut:

BAB I: PENDAHULUAN

Bab ini menjelaskan tentang latar belakang pengembangan aplikasi pencegah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging*, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika pembahasan.

BAB II: LANDASAN KEPUSTAKAAN

Bab ini menjelaskan tentang penelitian-penelitian sebelumnya yang terkait dengan pencegahan plagiarisme, teori-teori yang menjadi landasan dalam penelitian, dan penjelasan kakas bantu yang digunakan dalam penelitian.

BAB III: METODOLOGI PENELITIAN

Bab ini menjelaskan tahapan-tahapan penelitian yang dilakukan untuk mengembangkan aplikasi pencegah plagiarisme dengan menggunakan metode *snapshotting* dan *user activity logging*. Tahapan-tahapan tersebut meliputi studi literatur, rekayasa kebutuhan, perancangan, implementasi, pengujian, dan penarikan kesimpulan dan saran.

BAB IV: REKAYASA KEBUTUHAN

Bab ini menjelaskan proses rekayasa kebutuhan untuk membangun sistem pencegah plagiarisme dengan menggunakan metode *snapshotting* dan *user activity logging*. Tahapan rekayasa kebutuhan meliputi analisis kebutuhan, identifikasi aktor, spesifikasi dan manajemen kebutuhan, dan pemodelan kebutuhan. Kebutuhan tersebut meliputi kebutuhan fungsional dan non-fungsional. Pada tahapan ini juga dijelaskan deskripsi umum sistem.

BAB V: PERANCANGAN & IMPLEMENTASI

Bab ini menjelaskan proses perancangan dan implementasi dari aplikasi yang dibangun. Tahapan perancangan meliputi perancangan arsitektur, komponen, dan antarmuka. Tahapan implementasi meliputi implementasi kode, dan antarmuka.

BAB VI: PENGUJIAN

Bab ini menjelaskan proses pengujian perangkat lunak yang dibangun. Pengujian yang dilakukan berupa pengujian unit, pengujian integrasi, pengujian validasi, *automated testing*, dan pengujian *compatibility*.

BAB VII: PENUTUP

Bab penutup menjelaskan kesimpulan dari hasil penelitian yang telah dikerjakan, dan saran-saran untuk perbaikan penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian-penelitian sebelumnya pada bidang plagiarisme banyak berfokus pada *plagiarism detection*, baik menggunakan cara non-teknis seperti yang dilakukan Leung & Cheng (2017) dan McLafferty & Foust (2004), maupun menggunakan cara teknis dengan bantuan *plagiarism detection software* atau biasa disebut *plagiarism checker*. Berbeda dengan yang lain, Hellas, Leinonen, & Ihantola (2017) memfokuskan penelitiannya pada *plagiarism prevention* dengan studi kasus *take-home exams*.

Hellas, Leinonen, & Ihantola (2017) menggunakan metode pencegahan plagiarisme (*plagiarism prevention*) dengan bantuan perangkat lunak. Penelitian tersebut menggunakan kakas bantu *JPlag* dan algoritme *edit-distance*. *JPlag* digunakan untuk merekam waktu mulai tugas, waktu selesai tugas, dan alamat *IP* siswa. Algoritme *edit-distance* digunakan untuk menghitung nilai *edit-distance* pada berkas tugas. Rekaman waktu mulai dan waktu selesai digunakan untuk menemukan siswa yang mengerjakan pada waktu yang bersamaan, rekaman alamat *IP* digunakan untuk menemukan siswa yang mengerjakan pada tempat yang sama, dan analisis nilai *edit-distance* pada berkas tugas digunakan untuk menemukan siswa yang mendapatkan bantuan dari sumber eksternal atau tidak mengerjakan dengan kemampuannya sendiri.

Metode tersebut memiliki beberapa kelemahan, seperti tidak merekam semua aktivitas yang dilakukan selama mengerjakan tugas sehingga guru hanya menganalisis hasil akhir tugas dan mengabaikan proses penggerjaan tugas. Hal ini dapat dicurangi siswa dengan menyelesaikan tugas dengan membeli (Leung & Cheng, 2017). Analisis bantuan eksternal hanya dilakukan pada hasil akhir tugas sehingga kecurangan selama mengerjakan tugas dapat dicurangi dengan bantuan melalui *browser* maupun perangkat lunak sosial media. Proses rekaman dilakukan dengan bantuan perangkat lunak yang terikat dengan *platform* tertentu sehingga memiliki batasan bentuk tugas yang akan diberikan dan batasan pilihan kakas bantu yang akan digunakan siswa. Metode ini juga tidak memiliki *log* penggerjaan sehingga guru tidak dapat menilai usaha yang telah dilakukan siswa.

Penelitian yang diajukan akan menerapkan metode *screenshotting*, *user activity logging*, dan algoritme *edit-distance*. Metode *screenshotting* ditujukan untuk merekam semua perubahan berkas tugas sehingga guru dapat menganalisis keputusan, usaha, keterlibatan siswa dalam proses penggerjaan tugas, dan menemukan

personal siswa yang lemah. Metode *user activity logging* digunakan untuk merekam seluruh aktivitas siswa selama proses penggerjaan tugas. Algoritme *edit-distance* digunakan untuk menghitung nilai *edit-distance* pada berkas tugas. Perangkat lunak yang akan dibangun bersifat *platform agnostic* sehingga siswa mendapat kebebasan untuk menggunakan kakas bantu yang disukai dalam mengerjakan tugas, dan guru tidak memiliki batasan dalam memilih bentuk tugas yang akan diberikan.

2.2 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak atau *software engineering* memiliki definisi standar dari IEEE (1990) yang berbunyi “The systematic approach to the development, operation, maintenance and retirement of software”. Hal ini senada dengan pendapat Sommerville (2014) yang mendefinisikan rekayasa perangkat lunak sebagai suatu teknik disiplin yang fokus pada seluruh aspek dalam produksi perangkat lunak dari konsep awal hingga operasi dan pemeliharaan. Dalam hal ini Bell (2000) berpendapat bahwa tidak ada satu metode terbaik untuk melakukan rekayasa perangkat lunak, akan tetapi terdapat beberapa macam pendekatan-pendekatan tertentu.

Rekayasa perangkat lunak menjadi penting karena dua hal. Pertama karena semakin banyaknya aktivitas manusia yang bergantung pada perangkat lunak sehingga kita membutuhkan suatu cara untuk menghasilkan perangkat lunak yang handal dan terpercaya. Kedua karena penggunaan metode rekayasa perangkat lunak dapat membuat pengembangan perangkat lunak semakin murah, daripada menulisnya secara langsung tanpa menggunakan metode. Mengabaikan penggunaan metode rekayasa perangkat lunak dalam pengembangan perangkat lunak memiliki probabilitas terjadinya harga yang melambung untuk proses pengujian dan kesulitan dalam pemeliharaan jangka panjang (Sommerville, 2014).

Aplikasi metode rekayasa perangkat lunak yang benar dapat menghasilkan perangkat lunak yang baik. Perangkat lunak yang baik memiliki empat kriteria atau attribut, yaitu *acceptability*, *dependability* dan *security*, *efficiency*, dan *maintainability*. *Acceptability* merupakan karakteristik di mana perangkat lunak harus dapat diterima oleh penggunanya. *Dependability* merupakan karakteristik perangkat lunak yang memiliki sifat keamanan dan keandalan. Perangkat lunak tidak boleh menyebabkan kerugian fatal ketika terjadi kegagalan sistem. Atribut *efficiency* memungkinkan perangkat lunak berjalan tanpa menggunakan sumber daya yang ber-

lebihan atau boros. *Maintability* memungkinkan perangkat lunak mudah menerima perubahan (Sommerville, 2014).

2.3 Pengembangan Perangkat Lunak

Pengembangan perangkat lunak memiliki beberapa model pengembangan, arsitektur, pendekatan, dan pemodelan. Salah satu model pengembangan perangkat lunak adalah model *waterfall*. Model *waterfall* memiliki batasan di mana suatu tahapan harus diselesaikan terlebih dahulu sebelum melanjutkan ke tahapan selanjutnya. Salah satu arsitektur yang umum digunakan dalam pengembangan perangkat lunak berbasis *desktop* adalah *Model-View-controller*, arsitektur ini memisahkan bagian-bagian perangkat lunak sesuai dengan tanggung jawabnya. Dari sisi pendekatan terdapat beberapa pendekatan, salah satunya adalah pendekatan berorientasi objek yang di dalamnya terdapat aktivitas *object oriented analysis (OOA)*, dan aktivitas *object oriented design (OOD)*. Pengembangan perangkat lunak diakhiri dengan pengujian untuk memastikan perangkat lunak sudah sesuai dengan kebutuhan yang didefinisikan. Tahapan pengujian diantaranya adalah pengujian unit, integrasi dan validasi. Metode pengujian diantaranya adalah metode pengujian *black-box* dan *white-box*. Sedangkan teknik pengujian diantaranya adalah teknik pengujian *basis path*, *boundary value analysis* dan *equivalence partitioning*. Pengujian *compatibility* merupakan pengujian yang dilakukan untuk memastikan *compatibility* sistem, dan *automated testing* merupakan proses yang dilakukan untuk mempercepat pengujian.

2.3.1 Model Waterfall

Terdapat berbagai macam model dalam proses pengembangan perangkat lunak. Tidak ada model *universal* yang dapat digunakan untuk semua situasi. Model *waterfall* cocok digunakan untuk sistem yang permasalahan kebutuhannya telah dipahami dengan baik (Pressman, 2010). Model *waterfall* memiliki beberapa tahapan seperti pada Gambar 2.1, yaitu *requirement definition*, *system* dan *software design*, *implementation* dan *unit testing*, *integration* dan *system testing*, *operation* dan *maintenance*. Sesuai dengan rumusan masalah penelitian yang telah dipaparkan, tahapan yang dilakukan terbatas pada tahapan pengujian.

1. *Requirements analysis* dan *definition*

Pada tahapan ini kebutuhan, batasan, dan tujuan sistem ditentukan dengan hasil konsultasi bersama pengguna sistem. Ketiga data tersebut kemudian didefi-

nisikan lebih mendetail dan dijadikan sebagai spesifikasi sistem. Terdapat beberapa tahapan dalam proses rekayasa kebutuhan, diantaranya adalah elisitasi atau analisis, spesifikasi, manajemen, dan pemodelan. Tahapan analisis kebutuhan dilakukan untuk menggali kebutuhan, seperti bertanya kepada pengguna, dan menentukan objek dari sistem yang ingin dibangun. Spesifikasi kebutuhan dilakukan untuk menjelaskan kebutuhan secara lebih mendetail dengan jelas, tidak ambigu, mudah dipahami, dan konsisten. Manajemen kebutuhan dilakukan untuk mengidentifikasi, mengontrol, dan melacak kebutuhan, dan pemodelan kebutuhan dilakukan untuk menggambarkan kebutuhan secara visual maupun tekstual (Pressman, 2010).

2. *System dan software design*

Tahapan ini mengalokasikan kebutuhan sistem terhadap perangkat lunak maupun perangkat keras dan membentuk arsitektur sistem secara keseluruhan. Pada tahapan ini dilakukan identifikasi dan penjelasan pokok sistem perangkat lunak dan hubungannya satu sama lain. Di dalam *design* atau perancangan sistem terdapat beberapa tahapan, diantaranya adalah perancangan arsitektur, komponen, dan antarmuka. Perancangan arsitektur dilakukan untuk merancang desain arsitektur yang merepresentasikan struktur data dan komponen program yang dibutuhkan untuk membangun sistem. Perancangan komponen dilakukan untuk merancang struktur data, algoritme, dan mekanisme komunikasi antar komponen dalam sistem. Perancangan antarmuka dilakukan untuk merancang media yang efektif untuk komunikasi antar manusia dan komputer (Pressman, 2010).

3. *Implementation dan unit testing*

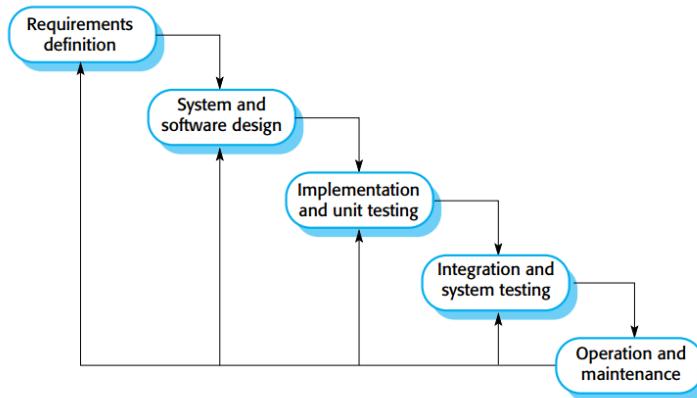
Tahapan ini merupakan aktivitas implementasi sistem. Sistem dibangun sebagai *unit-unit* yang kemudian diverifikasi agar memenuhi kriteria spesifikasi yang telah ditentukan. Tahapan implementasi dilakukan untuk mengimplementasikan tahapan-tahapan pada proses perancangan sebelumnya. Sementara itu, terdapat beberapa tahapan dalam proses pengujian, diantaranya pengujian unit, integrasi, validasi, dan pengujian sistem (Pressman, 2010).

4. *Integration dan system testing*

Pada tahapan ini *unit-unit* yang telah dibangun diintegrasikan dan diuji sebagai sistem yang lengkap, kemudian diserahkan kepada pengguna.

5. *Operation dan maintenance*

Tahap ini merupakan tahap pemeliharaan perangkat lunak, di mana di dalamnya terdapat aktivitas berupa perbaikan galat yang terdapat pada perangkat lunak, peningkatan, dan penyempuran perangkat lunak.



Gambar 2.1 Model waterfall (Sommerville, 2014)

2.3.2 Arsitektur *Model View Controller*

Model-View-Controller (MVC) merupakan sebuah model infrastruktur yang memisahkan antara *user interface* dari fungsionalitas inti sistem. *Model* memuat segala *content* dan fungsionalitas yang spesifik pada sistem yang dibangun, *processing logic* dan akses ke sumber informasi atau data eksternal. *View* memuat segala fungsi spesifik *interface*, dan segala proses fungsionalitas yang dibutuhkan *end-user*. *Controller* mengatur akses ke *model* dan *view* serta mengatur aliran data antara keduanya (Pressman, 2010). Penggunaan arsitektur *MVC* akan menghasilkan tiga *class* utama, yaitu *class view*, *model*, dan *class controller*. *Class model* akan memuat seluruh fungsionalitas inti yang spesifik pada sistem. Hal ini memberikan banyak keuntungan tatkala sistem akan dikembangkan lebih lanjut, karena pada pengembangan selanjutnya pengembang cukup fokus pada bagian *controller* dan *view*.

2.3.3 Pendekatan Berorientasi Objek

Langkah awal yang signifikan terhadap desain berorientasi objek terlahir dalam komunitas pengguna bahasa pemrograman *Ada*. Hal ini bermula ketika Grady Booch merasionalkan metode yang diciptakan Russell J. Abbott pada tahun 1983. Grady Booch kemudian menyebut karyanya sebagai *object-oriented design*. Keduanya merekomendasikan bahwa untuk memulai membangun desain berorientasi objek, dimulai dari deskripsi informal dari sistem yang akan dibangun. Dari deskripsi informal tersebut pengembang sistem dapat mengidentifikasi *class* dan objek dari kata benda dan operasi dari kata kerja yang ditemukan. Beberapa waktu setelahnya Grady Booch mengemukakan konsep baru yang tidak lagi berbasis

pada deskripsi naratif melainkan menggabungkan *object-oriented design* dengan metodologi yang sudah ada, kemudian menyebutnya sebagai *object-oriented development* (Capretz, 2003).

Pada tahun 1990-an Grady Booch bersama dengan dua orang rekannya, yaitu James Rumbaugh dan Ivar Jacobson mulai mengerjakan suatu metode yang diharapkan bersifat universal. Metode tersebut dibangun berdasarkan kombinasi dari fitur-fitur terbaik pada metode-metode *object-oriented analysis* dan *design method* yang ada. Metode ini juga mengadopsi fitur-fitur yang diajukan oleh ahli lain dalam rekayasa perangkat lunak. Karya ketiga orang tersebut yang dikenal sekarang dengan *Unified Modeling Language (UML)*. *UML* berisi notasi-notasi yang dapat digunakan untuk pemodelan dan pengembangan sistem berorientasi objek. Tujuh tahun setelahnya, yaitu pada tahun 1997 *UML* menjadi standar *de facto* untuk membangun perangkat lunak berorientasi objek (Pressman, 2010).

Membangun perangkat lunak dengan pendekatan berorientasi objek memiliki aktivitas *object oriented analysis (OOA)* yang di dalamnya terdapat proses eksplorasi *class* atau *object* dari pernyataan kebutuhan. Selanjutnya terdapat aktivitas *object oriented design (OOD)* yang di dalamnya terdapat proses *refine* dan *extend class* yang didapatkan dari *OOA*. Dalam pengembangan ini umumnya terdapat *boundary class* dan *controller class*. *Boundary class* bertindak sebagai pembatas atau *interface* antara pengguna dan sistem. *Boundary class* bertanggung jawab mengatur cara sebuah entitas *object* direpresentasikan kepada pengguna. *Controller class* mengatur pembuatan atau pembaharuan pada entitas *object*, penghubung antar *boundary* dan data yang didapatkan, dan komunikasi antar *object* (Pressman, 2010).

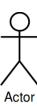
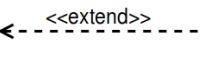
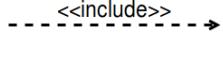
2.3.4 Pemodelan Berorientasi Objek

Notasi yang digunakan untuk pemodelan berorientasi objek adalah *Unified Modeling Language (UML)*. *UML* merupakan standar *de facto* dalam pemodelan berorientasi objek. Terdapat beberapa diagram inti dalam *UML*, diantaranya adalah *use case diagrams*, *sequence diagrams* dan *class diagrams*. *Use case diagrams* menunjukkan interaksi antar pengguna dan sistem secara visual. *Sequence diagrams* menunjukkan interaksi antar komponen atau objek di dalam sistem. *Class diagrams* menunjukkan struktur sistem dan relasi antar komponen pada sistem (Sommerville, 2014).

2.3.4.1 Use Case Diagram

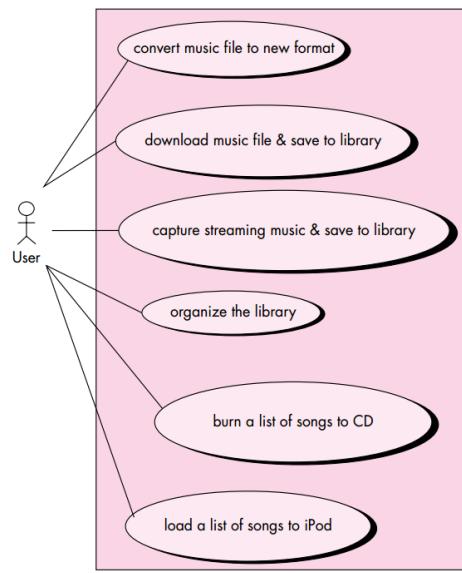
Use case diagram merupakan sebuah diagram statis yang menggambarkan interaksi antara pengguna dan sistem secara visual. *Use case diagram* membantu menentukan fungsionalitas sebuah sistem dari sudut pandang pengguna (Pressman, 2010). *Use case diagram* tidak menggambarkan tahapan. *Use case* digambarkan dengan bentuk oval dan aktor digambarkan dengan gambar orang (*stick figure*). Notasi *use case diagram* dan deskripsinya dipaparkan pada Tabel 2.1, dan contoh *use case diagram* terdapat dalam Gambar 2.2.

Tabel 2.1 Notasi *use case diagram* (Rumbaugh, Jacobson, & Booch, 2004)

Notasi	Deskripsi
 Actor	Notasi aktor menggambarkan pengguna atau sistem luar yang berinteraksi dengan sistem.
	Notasi <i>use case</i> merepresentasikan fungsionalitas yang dapat dijalankan oleh aktor.
	Notasi asosiasi merupakan jalur komunikasi antara aktor dan <i>use case</i> .
	Notasi <i>extends</i> merupakan sisipan fungsionalitas tambahan terhadap <i>base use case</i> yang tidak mengetahui sisipan tersebut.
	Notasi <i>include</i> merupakan sisipan fungsionalitas tambahan terhadap <i>basecase</i> yang secara eksplisit menjelaskan sisipan tersebut.
	Notasi <i>use case generalization</i> merupakan relasi antara <i>general use case</i> dan <i>specific use case</i> yang mewarisi dan menambahkan fitur padanya.

2.3.4.2 Sequence Diagram

Sequence diagram merupakan sebuah diagram dinamis yang menggambarkan interaksi antar objek di dalam sistem pada saat sistem mengeksekusi sebuah tugas. Terjadi pertukaran *messages* antar objek pada sistem untuk menyelesaikan

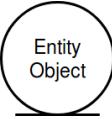
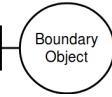
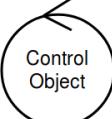


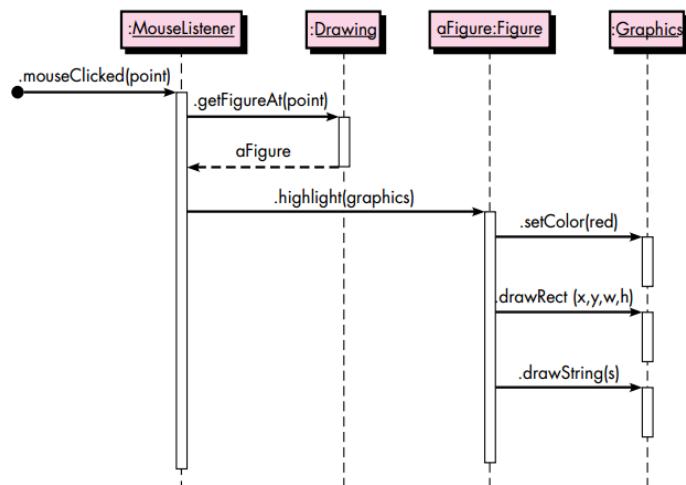
Gambar 2.2 Contoh *use case diagram* (Pressman, 2010)

sebuah tugas (Pressman, 2010). Notasi *sequence diagram* dan deskripsinya terdapat pada Tabel 2.2, dan contoh *sequence diagram* terdapat dalam Gambar 2.3.

Tabel 2.2 Notasi *sequence diagram*

Notasi	Deskripsi
	Notasi <i>lifeline</i> menggambarkan waktu aktif (<i>life</i>) suatu objek.
:Object 	Notasi <i>object</i> merepresentasikan objek yang berinteraksi dengan objek lainnya.
[]	Notasi <i>activation bar</i> merepresentasikan lama suatu objek aktif berinteraksi.
----- dispatch ----->	Notasi <i>method invocation</i> merepresentasikan suatu <i>method</i> dari sebuah objek yang panggil oleh objek lainnya.
<----- return ----->	Notasi <i>return</i> merepresentasikan data yang dikembalikan oleh method <i>caller</i> kepada method <i>caller</i> .
frame 	Notasi <i>frame</i> menggambarkan lingkup suatu interaksi dengan interaksi lainnya atau lingkup suatu interaksi dan dunia luar.

	Notasi <i>entity object</i> merepresentasikan objek yang bertanggung jawab untuk mengatur data dalam sebuah sistem.
	Notasi <i>boundary object</i> merepresentasikan objek yang bertanggung jawab untuk mengatur alur sebuah objek direpresentasikan kepada aktor.
	Notasi <i>controller object</i> merepresentasikan suatu objek yang mengatur pembuatan atau pembaharuan entitas <i>object</i> , penghubung antar <i>boundary</i> dan <i>entity</i> dan komunikasi antar <i>object</i> .



Gambar 2.3 Contoh *sequence diagram* (Pressman, 2010)

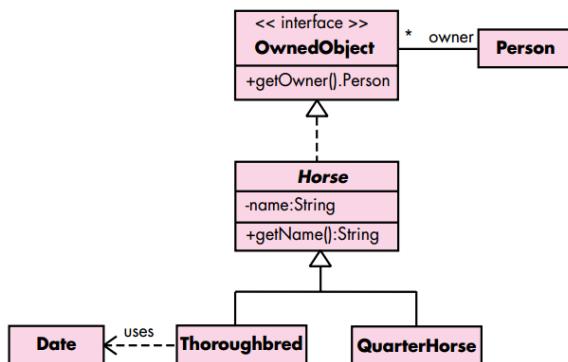
2.3.4.3 Class Diagram

Class diagram merupakan sebuah diagram statis atau *structural view* yang memodelkan *class*, *class attribute*, *class operation*, hubungan, dan asosiasi antar *class*. Elemen utama dari *class diagram* adalah kotak yang terbagi menjadi beberapa bagian. Bagian atas merepresentasikan nama *class*, bagian tengah merepresentasikan *attribute*, dan bagian bawah merepresentasikan *operation*. *Attribute* merupakan apa yang sebuah *class* atau objek ketahui, *operation* merupakan apa yang sebuah *class* atau objek dapat lakukan (Pressman, 2010). Notasi dan deskripsi

psi dari notasi tersebut dipaparkan pada Tabel 2.3, dan Contoh dari *class diagram* terdapat dalam Gambar 2.4.

Tabel 2.3 Notasi *class diagram*

Notasi	Deskripsi
	Notasi <i>class</i> merepresentasikan suatu <i>class</i> yang berisikan <i>attribute</i> dan <i>method</i> .
	Notasi <i>association</i> menggambarkan hubungan antar suatu <i>class</i> dengan <i>class</i> lainnya.
	Notasi <i>composition</i> menggambarkan hubungan kuat antar suatu <i>class</i> dengan <i>class</i> lain, di mana keduanya saling membutuhkan secara kuat dan saling membutuhkan.
	Notasi <i>aggregation</i> menggambarkan hubungan lemah antar suatu <i>class</i> dengan <i>class</i> yang lain, di mana keduanya tidak saling membutuhkan atau keduanya bisa berdiri sendiri.
	Notasi <i>generalization</i> merupakan relasi antara <i>parent class</i> dan <i>child class</i> , di mana <i>child class</i> mewarisi dan menambahkan fitur dari <i>parent class</i> .



Gambar 2.4 Contoh *class diagram* (Pressman, 2010)

2.3.5 Pengujian Perangkat Lunak

Penggunaan perangkat lunak yang masif tanpa adanya standardisasi membuat *IEEE* mempublikasikan sebuah *issue* pada bulan november hingga desember di tahun 1999. *IEEE* dan *ACM* kemudian membentuk tim gabungan untuk mendefinisikan standardisasi pada proses pengembangan perangkat lunak. Standardisasi tersebut memuat tentang *scientific principles, engineering processes, standards, methods, tools, measurement* dan *best practices*, proses pengujian atau *testing* termasuk di dalamnya (Burnstein, 2006).

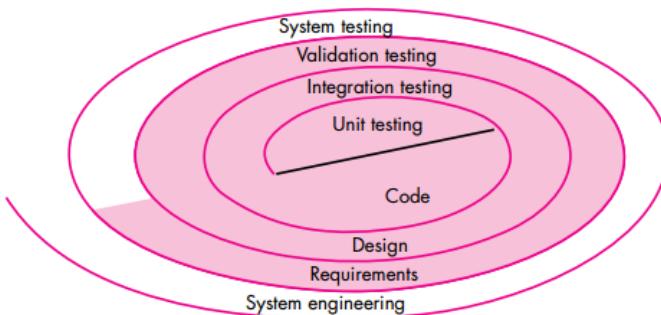
Pengujian bertujuan untuk menemukan cacat dan menguji kualitas suatu perangkat lunak. Dalam proses pengujian terdapat beberapa tahapan, metode, dan teknik. Setiap komponen pada bagian-bagian tersebut memiliki kelemahan dan kekurangan tertentu. Proses testing juga akan menghasilkan artefak-artefak seperti *test-scenario, test-case, test-data* dan *test scripts* (Burnstein, 2006).

2.3.5.1 Tahapan Pengujian

Terdapat beberapa tahapan dalam pengujian. Tahapan-tahapan tersebut diantaranya *unit testing, integration testing*, dan *validation testing* (Presman, 2010). Tahapan awal dalam pengujian adalah tahapan pengujian *unit*. Terdapat beberapa pendapat berbeda mengenai “*unit*” dalam *unit testing*. Fowler (2018) berpendapat bahwa definisi *unit* dapat berarti *single method, single class* maupun kumpulan dari beberapa *class*. Osherove (2015) tidak mengikat definisi *unit* pada *class* maupun *method*, melainkan mengikatnya dengan definsi atribut seperti *performance, reliability* dan *consistency*. Google tidak menggunakan istilah *unit, integration* ataupun *system* untuk merujuk pada tingkatan tertentu, melainkan menggunakan istilah *small, medium, large* (Whittaker, Arbon, & Carollo, 2012). Di sisi lain, Presman (2010) berpendapat bahwa fokus pengujian *unit* pada perangkat lunak konvensional adalah sebuah modul dan fokus pengujian *unit* pada perangkat lunak berorientasi objek adalah *class* dengan *testable unit* terkecil adalah *operation* atau *method*. Tahapan selanjutnya adalah *Integration testing*. *Integrartion testing* merupakan pengujian yang dilakukan pada kumpulan *unit* yang telah diuji. Hal ini dilakukan untuk memastikan suatu *unit* akan berjalan dengan baik jika dijalankan bersamaan dengan *unit* yang lain. Meskipun setiap *unit* telah diuji secara individu, terdapat beberapa kemungkinan galat yang terjadi ketika *unit-unit* tersebut diuji secara bersamaan. Beberapa kemungkinan galat yang akan terjadi di antaranya adalah hilangnya data ketika melintasi *interface*, dan komponen yang tidak berjalan semestinya ketika digabungkan. Pada proses pengujian *unit* maupun *integrar-*

tion, komponen tidak berupa *stand-alone program*. Maka dibutuhkan pembuatan *stub* ataupun *driver* selama proses pengujian. *Driver* hanyalah *main program* tiruan yang menjalankan kasus uji, sedangkan *stub* merupakan sebuah *dummy subprogram* yang bekerja menggantikan komponen-komponen pengujian yang asli.

Pada sistem dengan arsitektur berorientasi objek pengujian integrasi dengan pendekatan *incremental top-down integration* maupun *incremental bottom-up integration* tidak dapat digunakan, karena sistem berorientasi objek tidak memiliki kontrol hierarki yang jelas. Maka dibutuhkan pendekatan yang berbeda, yaitu dengan pendekatan integrasi *thread-based testing* dan *use-based testing*. *Thread-based testing* bekerja dengan mengintegrasikan sekumpulan *class* yang dibutuhkan untuk *input* tertentu. Setiap *thread* diuji dan diintegrasikan secara individu. *Use-based testing* berkerja dengan menguji *class* independen yang memiliki ketergantungan paling kecil dengan *class* lainnya. Setelah *class-class* tersebut diuji, lapisan *class* selanjutnya (*dependent classes*) yang menggunakan *independent class* diuji hingga seluruh sistem selesai diuji (Pressman, 2010). Tahapan terakhir adalah pengujian validasi atau *validation testing*. Pengujian validasi fokus terhadap *user-visible action* dan *user-recognizable output* dari sistem, atau dengan kata lain pengujian validasi menguji apakah sistem yang dibangun sudah sesuai dengan seluruh skenario kebutuhan yang didefinisikan (Pressman, 2010). Tahapan pengujian terlihat dalam Gambar 2.5.



Gambar 2.5 Tahapan pengujian (Pressman, 2010)

2.3.5.2 Metode Pengujian

Metode pengujian atau *point of view* dalam pengujian merupakan sudut pandang seorang *tester* tatkala mendesain sebuah *test case*. Terdapat di antaranya *white-box testing* dan *black-box testing*. Kedua metode tersebut memiliki kekurangan masing-masing. Metode *white-box testing* tidak dapat menemukan kebutuhan yang belum diimplementasikan (Dijkstra, 1970). Hal ini dapat diselesaikan dengan penggunaan metode *black-box testing*. Begitu juga dengan kekurangan

metode *black-box testing* yang dapat menghasilkan *test-case* yang tidak tepat dan tidak dapat menemukan bagian yang belum diuji (Savenkov, 2008). Kelemahan pada metode *black-box testing* ini dapat diselesaikan dengan menggunakan metode *white-box testing*. Oleh karena itu, penggunaan kedua metode dalam pengujian sistem menyelesaikan semua masalah pada masing-masing kekurangan yang dimiliki kedua metode.

White-box testing merupakan suatu metode pengujian di mana proses pengujinya dilakukan dengan melihat internal perangkat lunak. Maka pada tahapan ini seorang penguji wajib memiliki kemampuan pemrograman. Pengujian ini dilakukan untuk menguji apakah *logic* dan *data* pada perangkat lunak berfungsi sebagaimana mestinya (Myers, Sandler, & Badgett, 2011). Di sisi lain *black-box testing* hanya menguji bagian luar suatu perangkat lunak. Pada tahapan ini fungsionalitas suatu perangkat lunak diuji tanpa harus mengetahui internalnya. Oleh karena itu, tidak dibutuhkan kemampuan pemrograman saat membuat *test case* pada pengujian *black-box*. Penguji hanya mengetahui bagaimana seharusnya perangkat lunak berjalan (*what*), bukan bagaimana perangkat lunak itu melakukan sesuatu (*how*). Pada tahapan ini penguji hanya mempertimbangkan masukan (*input*) dan keluaran (*output*) perangkat lunak selama mendesain *test-case* (Myers, Sandler, & Badgett, 2011).

2.3.5.3 Teknik Pengujian

Terdapat beberapa teknik dalam pengujian. Pengujian semua *path* atau *rigorous testing* pada proses *white-box testing* tidak mungkin dilakukan sehingga digunakan teknik *basis path testing* untuk menentukan *path* yang akan diuji (Gregory, 2007). *Rigorous testing* atau *exhaustive testing* ($C\infty$) atau menguji semua kemungkinan juga tidak mungkin dilakukan pada saat pengujian *black-box* sehingga penggunaan teknik *equivalence partitioning* untuk menentukan masukan yang *valid* dan *invalid* diperlukan. Teknik *boundary value analysis* menjadi pelengkap teknik *equivalence partitioning*. *Boundary value analysis* digunakan untuk mendapatkan nilai masukan yang berada pada *boundary* atau *corner*, karena nilai-nilai pada bagian tersebut memiliki kemungkinan galat yang besar (Presman, 2010). Maka tiga teknik utama dalam pengujian perangkat lunak adalah:

1. *Basis path testing*
2. *Equivalence partitioning*
3. *Boundary value analysis*

Basis path testing merupakan suatu teknik pengujian pada metode *white-box* yang diajukan oleh McCabe pada tahun 1996. Teknik ini menggunakan penelitian McCabe yang sebelumnya yaitu *cyclomatic complexity* yang ia temukan pada tahun 1976. Meski saat ini *cyclomatic complexity* banyak dikaitkan secara erat dengan rumus untuk menentukan jumlah *independent path*. *Cyclomatic complexity* yang awalnya ditemukan pada tahun 1976 diajukan untuk mengukur tingkat kompleksitas suatu modul dalam sebuah program. Modul yang melebihi nilai *cyclomatic complexity* sepuluh direkomendasikan untuk dipisah atau dilakukan *refactoring*. Tujuan kedua dari rumus tersebut adalah untuk melihat tingkat terstrukturnya (*structuredness*) suatu modul. Hal yang membuat *cyclomatic complexity* dikaitkan secara erat dengan proses *basis path testing* karena kemudian McCabe sadar bahwa *cyclomatic complexity* dapat menemukan *independent path* pada suatu modul. *Independent path* adalah suatu jalur dalam sebuah program yang memperkenalkan setidaknya suatu rangkaian pemrosesan baru atau kondisi baru. Hal tersebut yang mengantarnya pada pengajuan *basis path testing* pada tahun 1996. *Basis path testing* memiliki keunggulan karena $V(G)$ atau *vector space* memiliki nilai yang menjadi batas atas *upper bound* untuk membuat *test-case* pada suatu program. Hal ini menguntungkan penguji karena dapat menentukan ukuran batas selesainya suatu pengujian. Selain itu, *basis path testing* memiliki nilai yang lebih baik dari pada *branch coverage*, yaitu $\text{branch coverage} \leq \text{cyclomatic complexity} \leq \text{all of paths}$ (Gregory, 2007). Langkah-langkah pengujian *basis path* adalah sebagai berikut (Presman, 2010):

1. Membuat *flow graph* dari desain atau *code*.

Flow graph memiliki beberapa notasi seperti *sequence*, *if*, *while*, *until* dan *case*. Gambar notasi-notasi tersebut terlihat pada Gambar 2.6. Terlihat *flow graph* pada Gambar 2.7 yang dihasilkan dari *pseudocode* 2.1.

2. Menentukan *cyclomatic complexity* dari *flow graph* yang dihasilkan.

Cyclomatic complexity adalah suatu metrik perangkat lunak yang memberikan ukuran kuantitatif kompleksitas logis dari suatu program. *Cyclomatic complexity* dapat ditentukan dengan menggunakan rumus 2.1:

$$\begin{aligned} V(G) &= \text{jumlah region} \\ V(G) &= E - N + 2 \\ V(G) &= P + 1, \text{ dimana } P = \text{predicate node} \end{aligned} \tag{2.1}$$

Maka Perhitungan *cyclomatic complexity* yang dilakukan pada *flow graph* 2.7 menghasilkan nilai sebagai berikut:

$$\begin{aligned}
 V(G) &= 2 \text{ regions} \\
 V(G) &= 4 \text{ edges} - 4 \text{ nodes} + 2 = 2 \\
 V(G) &= 1 \text{ predicate node} + 1 = 2
 \end{aligned} \tag{2.2}$$

Jadi, *flow graph* pada Gambar 2.7 memiliki nilai *cyclomatic complexity* = 2.

3. Menentukan *independent path*.

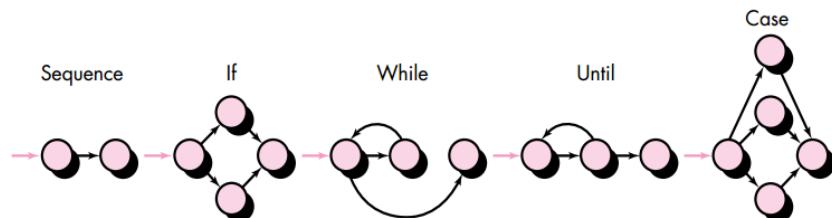
Independent path adalah suatu jalur dalam sebuah program yang memperkenalkan setidaknya suatu rangkaian pemrosesan baru atau kondisi baru. Nilai dari $V(G)$ memberikan batas atas dari banyaknya *independent path* dari sebuah program. Dari contoh *pseudocode procedure hallo* kita mendapatkan 2 jalur *independent* sebagai berikut:

Jalur 1: 1 - 2 - 4

Jalur 2: 1 - 3 - 4

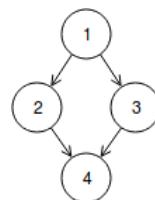
4. Membuat *test case* yang akan menjalankan setiap jalur pada *basis path*.

Terdapat dua *test case* yang akan dihasilkan. Pertama *test case* yang harus melewati jalur 1 sehingga pada *test case* pertama nilai *variable nama* harus bernilai *true*. Pada *test case* kedua nilai *variable nama* harus bernilai *false* agar jalur 2 dijalankan. *Test case* yang dihasilkan tampak seperti pada Tabel 2.4.



Gambar 2.6 Notasi *flow graph* (Presman, 2010)

Tabel Kode 2.1 Contoh *pseudocode*



Gambar 2.7 Representasi *control flow graph* dari pseudocode 2.1

Tabel 2.4 Test case dari contoh pseudocode 2.1

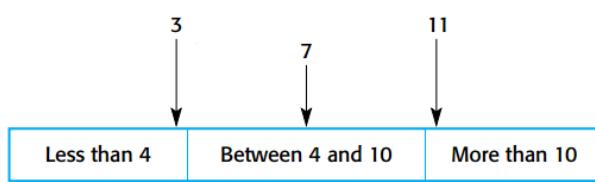
Jalur	Prosedur Uji	Expected Result
1	Memberikan nilai “Budi” pada variable <i>nama</i>	Program menampilkan “Hai Budi”
2	Tidak memberikan nilai apapun pada variable <i>nama</i>	Program menampilkan “Nama kosong”

Equivalence partitioning merupakan suatu teknik dalam metode pengujian *black-box* di mana prosesnya adalah membagi masukan kepada kelas-kelas data. Dari kelas-kelas tersebut *test case* nantinya didapatkan. *Equivalence class* merepresentasikan keadaan valid atau tidak validnya suatu masukan. Beberapa kondisi masukan di antaranya *numeric value*, *range of values*, *set of related values*, atau *boolean condition*. *Equivalence partitioning* dapat di definisikan sesuai dengan panduan berikut (Presman, 2010):

1. Jika kondisi masukan adalah *range*. Maka satu valid dan dua invalid *equivalence class* di definisikan.

2. Jika kondisi masukan adalah *specific value*. Maka satu valid dan dua invalid *equivalence class* di definisikan.
3. Jika kondisi masukan adalah *member of a set*. Maka satu valid dan satu invalid *equivalence class* di definisikan.
4. Jika kondisi masukan adalah *boolean*. Maka satu valid dan satu invalid *equivalence class* di definisikan.

Jika data masukan bertipe *range* dan masukan yang valid adalah *range* bernilai 4 hingga 10. Maka masukan yang tidak valid adalah semua angka yang kurang dari 4 dan semua angka yang lebih dari 10 seperti terlihat pada Gambar 2.8. Data yang digunakan untuk *test case* pada tipe masukan *range* berjumlah 3. Satu bernilai valid dan dua bernilai invalid. Oleh karena itu, data masukan *valid* yang kita gunakan adalah 11 dan 7. Sedangkan data *invalid* yang digunakan adalah 3.



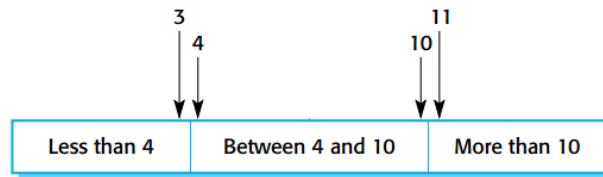
Gambar 2.8 Contoh *equivalence partitioning* (Sommerville, 2014)

Boundary value analysis merupakan suatu teknik yang melengkapi *equivalence partitioning*. *Boundary value analysis* dikembangkan untuk menganalisis nilai yang berada pada *boundary input*, karena nilai pada *boundary* atau *corner* memiliki peluang galat yang lebih tinggi. *Boundary value analysis* tidak berdiri sendiri, melainkan merupakan teknik yang melengkapi *equivalence partitioning*. Maka *boundary value analysis* mengambil nilai yang bersifat “pojok” dari nilai hasil *equivalence partitioning*. Panduan pemilihan nilai *boundary value analysis* di antaranya sebagai berikut (Presman, 2010):

1. Jika masukan merupakan sebuah *range* dari a hingga b. Maka nilai yang digunakan untuk *test case* adalah satu nilai di atas a dan satu nilai di bawah b.
2. Jika masukan merupakan sebuah *number of values*. Maka nilai yang digunakan untuk *test case* adalah nilai maksimum dan minimum serta satu nilai di atas maksimum dan satu nilai di bawah minimum.

Jika data masukan bertipe *range* dari 4 hingga 10. Maka nilai yang didapatkan untuk pengujian adalah nilai maksimum dan minimum serta satu nilai di

atas maksimum dan satu nilai di atas minimum. Jadi nilai yang didapatkan adalah 4, 10, 3, dan 11 seperti terlihat pada Gambar 2.9.



Gambar 2.9 Contoh *boundary value analysis* (Sommerville, 2014)

2.3.5.4 *Automated Testing*

Menjalankan semua *test case* secara manual sangat menyita waktu. Bahkan mereka dapat menyita 50% dari pada waktu pengembangan (Brooks Jr, 1995). *Automated testing* dapat digunakan untuk mempercepat proses pengujian. *Automated testing* adalah proses menjalankan *test case* secara otomatis. *Automated testing* berjalan dengan adanya *trigger*, baik dalam bentuk *event* maupun *time*. Prosesnya adalah dengan menjalankan *test case* dan membandingkannya dengan *output* yang sudah ditentukan. Hasil perbandingan tersebut akan dilaporkan kepada penguji secara otomatis setelah *automated testing* selesai dijalankan (Nshimiyiman, 2018). *Automated testing* dibangun dengan menggunakan *script* sesuai dengan *test case* yang didapatkan dari tahapan pengujian *white-box* maupun *black-box*. Tampak pada Tabel Kode 2.2 merupakan *script* yang dibangun untuk melakukan pengujian secara otomatis. *Script* tersebut didapatkan dari *test case* untuk pengujian *pseudo-code hallo* pada Tabel Kode 2.4. *Script* ini nantinya dijalankan dengan kakas bantu *Pytest* untuk melakukan pengecekan berhasil atau gagalnya suatu pengujian secara otomatis. Sedangkan *trigger* untuk menjalankan *Pytest* secara otomatis dilakukan dengan bantuan kakas bantu *Gitlab-CI*.

No	test_hallo
1	<pre>def test_case_1(): assert hallo("Budi") == "Hai Budi" 2 3 4 def test_case_2(): assert hallo("Ani") == "Nama Kosong"</pre>

Tabel Kode 2.2 Contoh *script automated testing*

2.3.5.5 Pengujian *Compatibility*

Pengujian *compatibility* adalah pengujian yang dilakukan pada sistem yang dibangun di atas beberapa media yang berbeda, seperti di atas *display devices*, *operating systems* dan *browser* yang berbeda-beda. Masalah *compatibility* yang kecil tidak memberikan dampak signifikan pada sistem, tetapi masalah yang besar dapat mengakibatkan dampak besar (Pressman, 2010). Terdapat berbagai macam *desktop environment* yang berbeda-beda pada sistem operasi *GNU/Linux*. Setiap *desktop environment* memiliki format nama *window* tersendiri. Sebuah sistem harus dapat menangani perbedaan format nama *window* agar dapat *compatible* antar *desktop environment*. Pengujian *compatibility* terhadap perbedaan format nama *window* antar *desktop environment* dilakukan dengan menjalankan sistem pada berbagai *desktop environment* yang berbeda. Pengujian berhasil jika sistem dapat menangani perbedaan format-format tersebut. Beberapa format nama *window* pada *desktop environment* di antaranya:

- “_NET_ACTIVE_WINDOW(WINDOW): window id # 0x3c00009”
- “_NET_ACTIVE_WINDOW(WINDOW): window id # 0x4a0000c, 0x0”

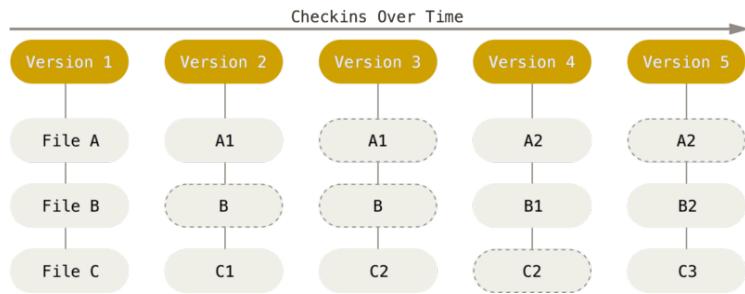
2.4 *Snapshotting*

Snapshot merupakan proses untuk merekam suatu keadaan media penyimpanan dalam suatu waktu tertentu. *Snapshot* yang dihasilkan dapat dikembalikan kapan pun pengguna butuhkan. Umumnya *snapshot* dapat dihasilkan dari beberapa kakas bantu seperti *data protection software*, *data analysis*, dan *replication application*. Implementasi untuk melakukan *snapshot* berbeda-beda, tergantung pada kebutuhan aplikasi(Garimella, 2018). *Version Control Software (VCS)* merupakan salah satu kakas bantu yang dapat digunakan untuk membuat *snapshot*. VCS lahir dimulai dengan adanya *Source Code Control System (SCCS)* pada tahun 1972 yang dibangun oleh Marc J. Rochkind, disusul kemunculan *Revision Control System (RCS)* pada tahun 1980, *Concurrent Versions System (CVS)* pada tahun 1985, kemudian *subversion* dan *git* pada tahun-tahun berikutnya (Ruparelia, 2010)

Git merupakan salah satu *version control software* yang digunakan secara umum. Hal ini disebabkan karena *git* memiliki sifat *distributed* sehingga memiliki kelebihan yang tidak dimiliki *version control* terpusat, salah satunya adalah kemampuan untuk melakukan seluruh operasi secara lokal (Ruparelia, 2010). *Snapshot* pada *git* disimpan di dalam *repository* dan digunakan untuk menyimpan suatu berkas dalam keadaan tertentu. Proses pembuatan *snapshot* dapat dilakukan

dengan meminta *git* untuk menambahkan semua berkas yang ada menggunakan perintah “*add*”. Proses selanjutnya dilakukan dengan menyimpan *snapshot* secara permanen dengan menggunakan perintah “*commit*”. Proses ini akan menyimpan *snapshot* secara permanen pada *database git*. Perintah lainnya yang dapat kita gunakan adalah “*log*” untuk melihat *history* dari *snapshot*, dan “*diff*” untuk melihat perbedaan antar *snapshot* (Chacon & Straub, 2014).

Umumnya *version control* beroperasi dengan menyimpan *diff* antar berkas pada suatu saat tertentu, cara ini biasa disebut dengan *delta-based version control*. Hal ini berbeda dengan *git* yang menyimpan keseluruhan *snapshot* pada suatu waktu tertentu. Setiap “*commit*” akan menyimpan seluruh keadaan berkas, kecuali berkas yang tidak berubah. *Git* melihat *project* layaknya *stream of snapshots*. Semua *snapshot* tersebut memiliki integritas yang berupa 40 karakter *hexadecimal* dihasilkan dari kalkulasi berkas pada suatu *snapshot* dengan algoritme *SHA-1*. Dengan *hash* tersebut, *git* dapat merujuk suatu *snapshot* tertentu (Chacon & Straub, 2014).



Gambar 2.10 Penyimpanan data sebagai *snapshot* dari waktu ke waktu (Chacon & Straub, 2014)

2.5 User Activity Logging

Log merupakan sebuah berkas yang memuat rekaman semua aktivitas yang terjadi pada perangkat lunak yang sedang berjalan. Pada umumnya *log* memiliki format berupa *plain text* dan memuat informasi berupa waktu, kategori dan deskripsi. *Log* secara umum digunakan pada *web server*, *operating system* dan berbagai macam perangkat lunak lain. Merekam aktivitas perangkat lunak maupun pengguna dan menyimpannya ke dalam *log* bertujuan untuk melakukan *debugging*, *monitoring*, *optimization*, *forensic analysis* dan beberapa tujuan lain. *Log* memiliki beberapa macam tipe, di antaranya *event logs*, *message logs* dan *transaction log* (DeLaRosa, 2018).

User activity logging adalah sebuah proses menyimpan aktivitas pengguna ke dalam sebuah *log*. *User activity* merupakan aktivitas pengguna dalam menggunakan perangkat lunak. *User activity* diantaranya adalah *keystrokes*, gerakan *mouse*, suara, gerakan, gerakan mata, tempat yang dikunjungi, berkas yang dibuka, dan aplikasi yang dijalankan (Macbeth, et al., 2007).

2.6 *Edit-distance Algorithm*

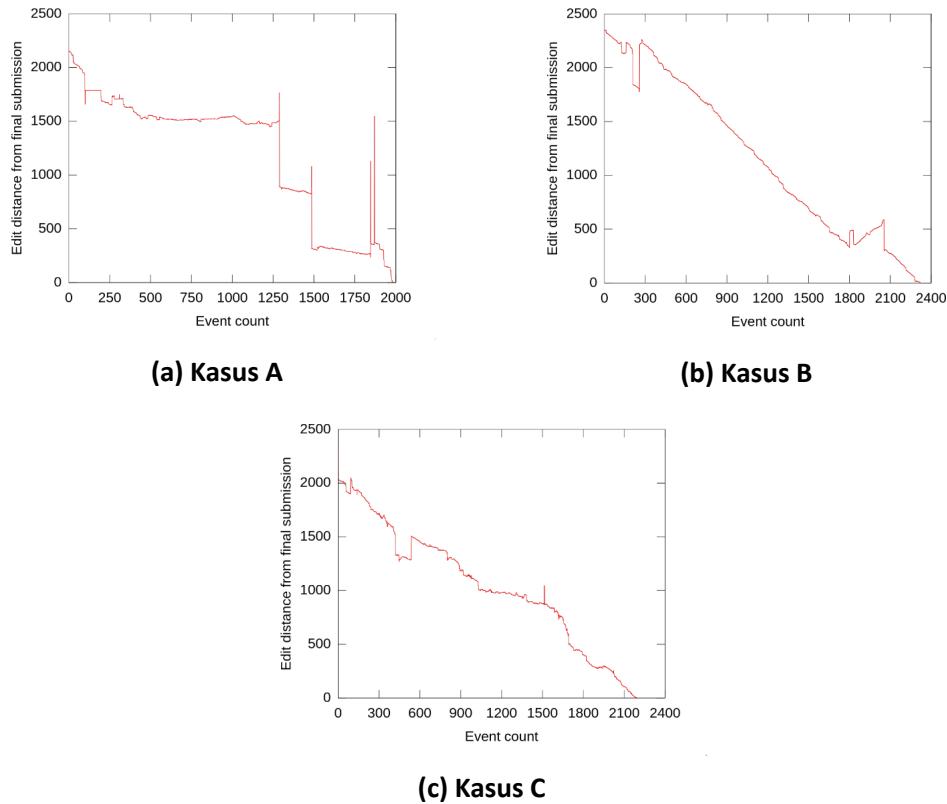
Edit-distance merupakan algoritme yang melakukan perhitungan pada penghapusan, sisipan, maupun substitusi antara dua buah kata. Algoritme ini lebih umum dikenal dengan *edit distance*, tetapi umumnya *edit distance* merujuk pada *levenshtein distance*. Algoritme ini ditemukan oleh matematikawan rusia, Vladimir Levenshtein pada 1965. Pada algoritme ini semua karakter yang ada, memiliki *unit cost* (harga yang nantinya dikalkulasi) kecuali substitusi yang dilakukan pada karakter itu sendiri. Secara umum algoritme ini memiliki nilai yang sama dengan nilai minimal operasi yang diperlukan untuk mengubah karakter “a” menjadi “b” (Navarro, 2001). Secara formal dapat ditulis sebagai:

$${}^w ins^{(x)}, {}^w del^{(x)}, {}^w sub^{(x,y)} \quad (2.3)$$

Contoh perhitungan dari algoritme ini, seperti perubahan dari kata “tekkom” menjadi “filkom”, yang memiliki nilai *edit-distance* berjumlah 3.

1. tekkom → fekkom (penggantian dari “t” ke “f”).
2. fekkom → fikkom (penggantian dari “e” ke “i”).
3. fikkom → filkom (penggantian dari “k” ke “l”).

Algoritme *edit-distance* nantinya dapat digunakan untuk mengukur kemirian proses penggerjaan tugas. Grafik akan dikalkulasi dari *snapshot* satu ke yang lain. Tampak pada Gambar 2.11a menunjukkan siswa yang mengerjakan tugasnya sendiri kemudian mengalami kesulitan, pada akhirnya ia meminta jawaban temannya kemudian ditempelkan pada tugasnya sendiri. Gambar 2.11b menunjukkan siswa yang meminta jawaban teman sejak awal, kemudian meniru jawaban tersebut tanpa melakukan salin-tempel, tetapi melakukan beberapa modifikasi di akhir. Gambar 2.11c menunjukkan siswa yang mengerjakan tugasnya sendiri dan terlihat tidak adanya proses salin-tempel selama penggerjaan tugas (Hellas, Leinonen, & Ihantola, 2017).



Gambar 2.11 Grafik *edit-distance*

2.7 Teknologi Pengembangan Perangkat Lunak

2.7.1 Python

Python adalah bahasa pemrograman dengan *syntax* yang elegan dan sederhana yang dikembangkan oleh Guido van Rossum pada tahun 1990. *Python* bersifat *interpreted* dan mendukung paradigma *object-oriented*. *Python* juga memiliki dukungan struktur data tingkat tinggi seperti *dictionaries* dan *list*. Berbeda dengan bahasa *scripting* yang lain, *Python* dapat digunakan secara gratis, bahkan untuk tujuan komersial (Sanner, et al., 1999).

2.7.2 Git (*Distributed Version Control*)

Distributed version control merupakan sebuah konsep di mana aplikasi *version control* dapat merekam semua perubahan yang terjadi pada sebuah berkas secara terdistribusi atau tidak terpusat. Sifatnya yang tidak terpusat memiliki kelebihan untuk melakukan semua operasi secara lokal. Berbeda dengan *version control* lainnya yang menggunakan *diff* untuk menyimpan berkas dari suatu waktu ke waktu, *git* menggunakan model *snapshot* atau merekam seluruh berkas pada suatu

waktu tertentu. Hal ini memiliki kelebihan seperti kecepatan dalam konstruksi berkas yang diminta pada waktu tertentu. Pada mulanya *git* dikembangkan oleh Linus Torvalds pada tahun 2005, Linus kemudian menunjuk Junio Hamano untuk menjadi *maintainer* hingga sekarang (Chacon & Straub, 2014). *Git* sejak awal diciptakan secara modern sehingga memiliki integritas menggunakan *hash SHA1* yang berupa 40 karakter heksadesimal. Hal ini yang dijelaskan oleh Linus Torvalds sebagai *content addressable* layaknya *filesystem* (Torvalds, 2018). Tabel Kode 2.3 menunjukkan perintah pada *git* untuk menyimpan perubahan suatu berkas.

No	commit
1	git commit -m "saved"

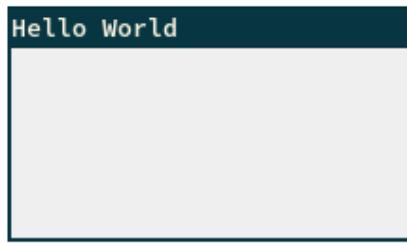
Tabel Kode 2.3 Perintah untuk menyimpan perubahan pada *git*

2.7.3 *Qt (Widget Toolkit)*

Qt merupakan pustaka yang digunakan untuk membangun *graphical user interface*. Awalnya *Qt* dikembangkan oleh Haavard Nord dan Eirik Chambe-Eng. Pada saat ini *Qt* dikembangkan oleh *The Qt Company* (*Qt History* 2018). *Qt* memiliki beberapa lisensi, di antaranya lisensi komersial dan lisensi *GPL 2.0*. Modul-modul yang dimiliki *Qt* di antaranya adalah *Qt Core*, *Qt GUI*, dan *Qt Widgets* (*Qt About* 2018). Secara umum *Qt* digunakan untuk membangun *graphical user interface* agar memudahkan interaksi pengguna dengan aplikasi. Terlihat pada gambar 2.12 *Qt* digunakan untuk membuat *window* sederhana dengan menggunakan kode pada Tabel Kode 2.4.

No	hello
1	<code>import sys</code>
2	<code>from PyQt5.QtWidgets import QApplication, QWidget</code>
3	
4	<code>if __name__ == '__main__':</code>
5	<code> app = QApplication(sys.argv)</code>
6	<code> window = QWidget()</code>
7	<code> window.resize(250, 150)</code>
8	<code> window.setWindowTitle('Hello World')</code>
9	<code> window.show()</code>
10	<code> sys.exit(app.exec_())</code>

Tabel Kode 2.4 Kode untuk membuat *window* sederhana



Gambar 2.12 *Window sederhana dengan Qt*

2.7.4 *Draw.io*

Draw.io merupakan sebuah kertas bantu untuk menggambar diagram atau pun *mock up*. *Draw.io* memiliki lisensi perangkat lunak bebas (*Apache v2*) dan dapat diakses melalui peramban secara daring. *Draw.io* dibangun oleh Gaudenz Alder menggunakan bahasa pemrograman *javascript* pada tahun 2000. Peramban yang didukung diantaranya *IE 11*, *Chrome 32+*, *Firefox 38+*, *Safari 9.1.x*, dan *Opera 20+* (*About draw.io* 2019).

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan langkah-langkah penelitian. Langkah pertama diawali dengan studi literatur. Langkah selanjutnya diikuti dengan rekayasa kebutuhan, perancangan, implementasi dan pengujian yang merupakan urutan tahapan pada pengembangan sistem yang menggunakan model *waterfall*. Pemilihan model *waterfall* dalam pengembangan sistem pada penelitian ini disebabkan kebutuhan sistem yang sudah dapat didefinisikan dengan baik pada tahap awal pengembangan sistem. Pengujian merupakan langkah terakhir dalam pengembangan sistem pada penelitian ini, hal ini sesuai dengan rumusan masalah penelitian. Langkah terakhir penelitian ditutup dengan penarikan kesimpulan dan saran. Langkah-langkah tersebut dapat dilihat pada pada Gambar 3.13.



Gambar 3.13 Diagram alir metodologi

3.1 Studi Literatur

Studi literatur dibutuhkan untuk mendalami teori-teori dan urgensi tentang plagiarisme lebih dalam. Selain itu, studi literatur juga dibutuhkan untuk mendalami teori rekayasa perangkat lunak untuk mengembangkan perangkat lunak pence-

gah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging*. Sumber dari literatur tersebut berasal dari jurnal, buku, situs resmi dan dokumentasi perangkat lunak yang digunakan. Daftar literatur yang dipelajari terkait dengan:

1. Penelitian-penelitian sebelumnya yang terkait dengan plagiarisme
2. Metode *user activity logging*, *screenshotting* dan *edit-distance algorithm*
3. Teori pengembangan perangkat lunak, yang di dalamnya meliputi teori rekayasa kebutuhan, perancangan, implementasi, dan pengujian
4. Teknologi yang digunakan dalam penelitian

3.2 Rekayasa Kebutuhan

Rekayasa kebutuhan merupakan kumpulan tugas dan teknik untuk memahami kebutuhan. Terdapat empat tahapan dalam rekayasa kebutuhan, yaitu analisis kebutuhan, spesifikasi kebutuhan, manajemen kebutuhan, dan pemodelan kebutuhan. Penjelasan tentang tahapan-tahapan yang dilakukan dalam rekayasa kebutuhan adalah sebagai berikut:

1. Analisis kebutuhan

Pada tahapan ini dilakukan penggalian kebutuhan sistem. Kebutuhan fungsional sistem didapatkan dari:

- (a) Metode yang telah digunakan pada penelitian (Hellas, Leinonen, & Ihantola, 2017).
- (b) “*Future work*” penelitian Hellas, Leinonen, & Ihantola (2017), dan penelitian Leung & Cheng (2017) yang memaparkan cara-cara plagiarisme di mana cara-cara tersebut belum dapat ditangani oleh penelitian Hellas, Leinonen, & Ihantola (2017).
- (c) Kelemahan dari penelitian Hellas, Leinonen, & Ihantola (2017) yang akan disempurnakan.

Pada sumber kebutuhan fungsional pertama didapatkan kebutuhan fungsional untuk merekam waktu mulai penggerjaan, waktu selesai penggerjaan, rekaman alamat *IP*, dan menghitung nilai *edit-distance* pada berkas tugas. Sumber kebutuhan fungsional kedua dan ketiga menghasilkan kebutuhan fungsional untuk merekam

perubahan berkas tugas dengan metode *snapshotting*, merekam aktivitas siswa menggunakan metode *user activity logging*, dan mengembangkan sistem dengan sifat *agnostic* agar tidak terikat pada *platform* tertentu. Komponen aktivitas siswa yang direkam adalah seluruh *window*, *window* yang sedang aktif, nama *login*, dan nama *device*.

Kebutuhan non-fungsional didapatkan dari batasan sistem yang hanya dapat berjalan pada sistem operasi *GNU/Linux*, terdapat berbagai macam *desktop environment* pada sistem operasi *GNU/Linux* sehingga sistem yang dibangun harus *compatible* dengan enam *desktop environment* yang umum digunakan. Keenam *desktop environment* tersebut adalah *GNOME*, *KDE*, *XFCE*, *Unity*, *Cinnamon* dan *Phanteon*.

2. Spesifikasi kebutuhan

Pada tahapan ini hasil dari analisis kebutuhan pada tahapan sebelumnya dielaskan secara lebih mendetail dengan jelas, tidak ambigu, mudah dipahami, dan konsisten. Kebutuhan fungsional mendeskripsikan apa yang dapat dilakukan oleh sistem yang dibangun dan kebutuhan non-fungsional mendeskripsikan batasan atau kualitas dari sistem yang dibangun.

3. Manajemen kebutuhan

Tahapan ini dilakukan untuk memudahkan dalam mengidentifikasi, mengontrol dan melacak kebutuhan. Maka pada tahapan ini dilakukan kodefikasi pada kebutuhan-kebutuhan yang ada. Kode LUP-F-1 dan LUPR-F-1 untuk kebutuhan fungsional, LUPR-NF-1 dan LUPR-NR-1 untuk kebutuhan non-fungsional.

4. Pemodelan kebutuhan

Tahap terakhir adalah pemodelan kebutuhan. Untuk menggambarkan fungsi dan batasan sistem secara visual, penelitian ini menggunakan alat bantu *use case diagram*, sedangkan untuk menjelaskan skenario interaksi antar aktor dan sistem secara mendetail, penelitian ini menggunakan alat bantu *use case scenario*. Penggunaan kedua diagram tersebut dalam tahapan ini dapat memodelkan kebutuhan sistem baik secara visual (*use case diagram*) maupun secara tekstual dengan detail (*use case scenario*).

3.3 Perancangan

Tahapan perancangan sistem menjelaskan tentang rancangan sistem yang akan dibangun berdasarkan hasil rekayasa kebutuhan pada tahapan sebelumnya.

Rancangan ini nantinya akan menjadi acuan pada tahap implementasi dan pengujian. Tahap-tahap yang dilakukan pada perancangan sistem adalah:

1. Perancangan arsitektur

Sistem akan dibangun dengan arsitektur *Model-View-Controller (MVC)*. Arsitektur *MVC* dipilih agar komponen fungsionalitas sistem terpisah dari komponen antarmuka sistem. Hal ini memudahkan pengembangan selanjutnya, karena pengembang cukup fokus pada masing-masing komponen yang terpisah. Arsitektur ini juga memudahkan proses *porting* maupun penggantian antarmuka, karena pengembang cukup fokus pada komponen antarmuka sistem. Untuk menggambarkan interaksi yang terjadi antar komponen pada sistem, penelitian ini menggunakan alat bantu *sequence digaram*. Terdapat tiga sampel *sequence diagram* yang akan dipaparkan pada tahapan ini. Untuk menggambarkan struktur sistem dan relasi antar komponen pada sistem, penelitian ini menggunakan alat bantu *class diagram*.

2. Perancangan komponen

Pada tahapan ini komponen sistem akan dirancang menggunakan *pseudocode*. *Pseudocode* inilah yang nantinya menjadi acuan pada tahapan implementasi. Pada tahapan ini terdapat tiga sampel perancangan komponen yang akan dipaparkan.

3. Perancangan antarmuka

Pada tahapan ini rancangan antarmuka sistem dan tata letak komponen antarmuka di dalamnya digambarkan dengan gambaran sederhana. Perancangan antarmuka dilakukan dengan kakas bantu *draw.io*. Terdapat tiga sampel perancangan antarmuka yang akan dipaparkan pada tahapan ini.

3.4 Implementasi

Pada tahapan implementasi, hasil perancangan pada tahapan sebelumnya diimplementasikan, terdapat dua tahapan dalam implementasi yaitu:

1. Implementasi kode program

Pada tahapan ini hasil rancangan komponen sistem yang berupa *pseudocode* diimplementasikan menjadi *working code* menggunakan bahasa pemrograman *Python*. Selain menggunakan *Python*, metode *screenshotting* yang diimplementasikan akan menggunakan bantuan kakas bantu *git*.

2. Implementasi antarmuka

Pada tahapan ini hasil rancangan antarmuka sistem diimplementasikan menggunakan *widget toolkit Qt*.

3.5 Pengujian

Pada tahapan pengujian, dilakukan pengujian terhadap sistem yang telah selesai diimplementasikan. Tahapan-tahapan yang akan dilakukan adalah:

1. Pengujian unit

Tahap pengujian unit dilakukan untuk memastikan hasil implementasi kode program sesuai dengan hasil perancangan komponen. Penelitian ini mengikuti pendapat Pressman (2010) terkait dengan pemahaman *unit*. Maka tahapan ini menguji setiap unit terkecil dari sistem yaitu sebuah *class*. *Class controller* dan *model* diuji hingga batas *code coverage* mencapai 100%, karena semua kode inti program terdapat pada kedua *class* tersebut. Berbeda dengan *class view* hanya memanggil fungsi-fungsi yang terdapat pada *class controller* dan *model* sehingga pada penelitian ini pengujian unit pada *class view* hanya diuji hingga *code coverage* mencapai 70%. Pengujian unit dilakukan menggunakan metode *white-box testing* dan kasus uji didapatkan dengan teknik *basis path testing*. Pengujian setiap unit pada tahapan ini dilakukan secara terisolasi sehingga dilakukan pembuatan *stub* atau “*dummy subprogram*”. Terdapat tiga sampel pengujian unit yang akan dipaparkan pada tahapan ini.

2. Pengujian integrasi

Setelah semua unit selesai diuji, maka dilakukan tahapan pengujian integrasi. Pengujian integrasi dilakukan untuk memastikan integrasi antar unit berjalan dengan baik. Maka pada pengujian integrasi, *function* pada suatu *class* yang memanggil *function* pada *class* lain tidak digantikan dengan *stub* atau *fake* sebagaimana ketika pada pengujian unit. Pada penelitian ini perangkat lunak yang dikembangkan menggunakan pendekatan dan pengembangan berorientasi objek sehingga pendekatan pengujian integrasi yang digunakan adalah *use-based testing*. Integrasi diawali dari *class* yang memiliki *dependent class* paling sedikit, yaitu dimulai dari *class model*, kemudian *class controller*, dan yang terakhir adalah *class view*. Pengujian integrasi menggunakan metode *white-box testing* dan teknik *basis path testing*. Terdapat satu sampel yang dipaparkan pada tahapan ini.

3. Pengujian validasi

Pengujian validasi menguji apakah sistem yang dibangun sudah sesuai dengan kebutuhan yang didefinisikan. Pengujian validasi fokus terhadap *user-visible action* dan *user-recognizable output* dari sistem (Pressman, 2010). Maka pada tahapan ini seluruh skenario kebutuhan diuji. Pengujian validasi dilakukan menggunakan metode *black-box testing*. Teknik yang digunakan untuk mendapatkan kasus uji dan data uji adalah teknik *equivalence partitioning* dan *boundary value analysis*. Kedua teknik tersebut digunakan untuk menghindari *rigorous testing* atau mencoba semua kemungkinan.

4. Automated testing

Automated testing dilakukan untuk meminimalisir waktu yang digunakan untuk pengujian. Pada tahapan ini dibangun *script* yang menjalankan kasus uji secara otomatis. Kasus uji didapatkan dari pengujian unit dan pengujian integrasi yang telah dilakukan pada tahapan sebelumnya. *Automated testing* berjalan secara otomatis jika terjadi perubahan pada berkas *source code* sistem. Penelitian ini menggunakan kaca bantu *Pytest* untuk membangun *script*, dan menggunakan *Gitlab-CI* untuk menjalankan *script* secara otomatis. Terdapat tiga sampel *automated testing* yang dipaparkan pada tahapan ini.

5. Pengujian compatibility

Pengujian *compatibility* merupakan pengujian terhadap kebutuhan non-fungsional sistem. Pada pengujian ini sistem *Lup Recorder* dan *Lup Viewer* dijalankan pada enam *desktop environment* yang umum digunakan pada sistem operasi *GNU/Linux*. Keenam *desktop environment* tersebut adalah *GNOME*, *KDE*, *XFCE*, *Unity*, *Cinnamon* dan *Phanteon*. Pengujian ini dilakukan untuk memastikan apakah sistem yang dibangun sudah *compatible* dengan berbagai macam *desktop environment*.

3.6 Penarikan Kesimpulan

Kesimpulan akan ditarik pada akhir penelitian. Penarikan kesimpulan dapat dilakukan setelah tahap analisis kebutuhan, perancangan, implementasi, dan pengujian selesai dilakukan. Kesimpulan akan ditarik berdasarkan dari analisis hasil sistem yang telah dibangun. Kesimpulan tersebut akan menjawab rumusan masalah yang telah dipaparkan dan memuat catatan untuk pengembangan sistem di waktu mendatang.

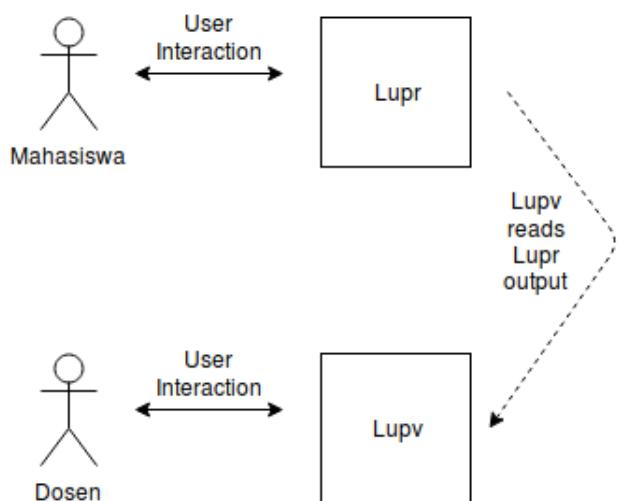
BAB 4 REKAYASA KEBUTUHAN

4.1 Deskripsi Umum Sistem

Sistem yang dikembangkan pada penelitian ini adalah sistem pencegah plagiarisme dengan menggunakan metode *screenshotting* dan *user activity logging*. Sistem ini dibangun dengan tujuan untuk meminimalisir adanya tindak plagiarisme. Sistem ini dibagi menjadi dua bagian, pertama yaitu *Lupv* (*Lup Viewer*) yang digunakan oleh dosen untuk membaca hasil rekaman yang dilakukan oleh sistem kedua. *Lupr* hanya dapat membaca berkas tugas yang berbentuk *plain text format*. Sistem kedua yaitu *Lupr* (*Lup Recorder*), sistem ini digunakan untuk merekam perubahan tugas dan aktivitas siswa selama penggeraan tugas berlangsung. *Lupv* digunakan oleh dosen sedangkan *Lupr* digunakan oleh mahasiswa.

Lupv memiliki beberapa fungsionalitas, di antaranya menampilkan seluruh daftar rekaman, melihat aktivitas yang dilakukan selama penggeraan, dan melihat perubahan berkas tugas dari waktu ke waktu. Di lain sisi, *Lupr* memiliki fungsionalitas di antaranya adalah merekam perubahan berkas tugas, alamat *IP*, *login name*, *device name*, seluruh *window* dan *window* yang aktif.

Cara kerja sistem ini dimulai dengan mahasiswa yang menjalankan *Lupr* pada berkas tugas yang diberikan dosen. Setelah tugas selesai dikerjakan, maka berkas tersebut kemudian dikirimkan ke dosen yang bersangkutan. Dosen yang bersangkutan kemudian dapat melihat seluruh hasil rekaman menggunakan *Lupv*.



Gambar 4.14 Gambaran umum sistem

4.2 Analisis Kebutuhan

Analisis kebutuhan pada sistem ini didapatkan dari beberapa sumber. Sumber pertama adalah metode penelitian yang telah dilakukan oleh Hellas, Leinonen, & Ihantola (2017). Sumber kedua didapatkan dari “*future work*” penelitian Hellas, Leinonen, & Ihantola (2017) dan penelitian Leung & Cheng (2017). Penelitian Leung & Cheng (2017) memaparkan cara-cara plagiarisme di mana cara-cara tersebut belum dapat ditangani oleh metode penelitian yang telah dilakukan Hellas, Leinonen, & Ihantola (2017). Sumber kebutuhan ketiga adalah kelemahan dari penelitian Hellas, Leinonen, & Ihantola (2017) yang akan disempurnakan. Ketiga sumber tersebut menghasilkan kebutuhan-kebutuhan pokok sistem yaitu mengimplementasikan metode *screenshotting* untuk merekam perubahan berkas, metode *user activity logging* untuk merekam aktivitas siswa saat mengerjakan tugas, serta membangun sistem dengan sifat *agnostic* agar tidak terikat dengan *platform* tertentu.

Sistem yang dikembangkan memiliki batasan hanya dapat berjalan pada sistem operasi *GNU/Linux*. Terdapat banyak *desktop environment* yang ada pada sistem operasi *GNU/Linux*. Maka sistem harus dapat berjalan dengan baik pada berbagai macam *desktop environment* yang memiliki format nama *window* yang berbeda-beda. Oleh karena itu, kebutuhan non-fungsional yang harus dimiliki oleh sistem adalah *compatibility* sistem pada berbagai macam *desktop environments*. Jumlah *desktop environment* yang tidak terbatas membuat penelitian ini membatasi jumlah *desktop environment* yang didukung. *Desktop environment* yang didukung terbatas pada enam *desktop environment* yang umum digunakan, yaitu *GNOME*, *KDE*, *Xfce*, *Unity*, *Cinnamon* dan *Phanteon*.

4.3 Identifikasi Aktor

Berdasarkan proses analisis kebutuhan yang dilakukan, terdapat satu aktor utama pada sistem *Lup Viewer* yaitu dosen, dan satu aktor utama pada sistem *Lup Recorder* yaitu mahasiswa. Tabel 4.5 memaparkan daftar aktor yang terlibat dalam sistem *Lup Viewer* beserta deskripsinya, dan Tabel 4.6 memaparkan daftar aktor yang terlibat pada sistem *Lup Recorder* beserta deskripsinya.

Tabel 4.5 Identifikasi aktor pada sistem *Lup Viewer*

No	Aktor	Deskripsi
1	Dosen	Dosen merupakan suatu individu atau kelompok yang berperan dalam memberikan tugas dan menganalisis hasil tugas yang telah dikerjakan oleh mahasiswa.

Tabel 4.6 Identifikasi aktor pada sistem *Lup Recorder*

No	Aktor	Deskripsi
1	Mahasiswa	Mahasiswa merupakan suatu individu atau kelompok yang berperan dalam mengerjakan tugas yang diberikan oleh dosen.

4.4 Kebutuhan Fungsional Sistem

Kebutuhan fungsional merupakan sebuah kebutuhan yang harus disediakan oleh sistem, bagaimana sistem bereaksi terhadap suatu masukan, dan bagaimana sistem berperilaku pada suatu keadaan (Sommerville, 2014). Daftar kebutuhan fungsional didapatkan dari proses analisis kebutuhan yang telah dilakukan sebelumnya. Dalam penelitian ini, kebutuhan fungsional memiliki sistem penomoran LUPV-F-XX-YY untuk sistem *Lup Viewer* dan LUPR-F-XX-YY untuk sistem *Lup Recorder*. *LUPV* dan *LUPR* merupakan nama sistem, F menunjukkan tipe kebutuhan yaitu kebutuhan fungsional, XX menunjukkan nomor kebutuhan, dan YY menunjukkan nomor spesifikasi kebutuhan. Tabel 4.7 dan Tabel 4.8 memaparkan daftar kebutuhan fungsional, beserta definisi, aktor yang terlibat, dan pemetaan nama *use case* yang didapatkan dari hasil analisis kebutuhan untuk sistem *Lup Recorder* dan *Lup Viewer*.

Tabel 4.7 Spesifikasi kebutuhan fungsional, definisi, dan pemetaan nama *use case* sistem *Lup Recorder*

No	Kode Kebutuhan	Deskripsi Kebutuhan	Aktor	Nama <i>Use Case</i>
1	LUPR-F-01	Sistem harus menyediakan fungsi untuk merekam pengerjaan tugas.	Mahasiswa	Merekam pengeraaan tugas.
1.1	LUPR-F-01-01	Sistem harus menyediakan tombol “ <i>Start Recording</i> ” sebagai sarana untuk menjalankan perekaman tugas.		
1.2	LUPR-F-01-02	Sistem harus menyediakan fungsi untuk memilih direktori tempat tugas dikerjakan.		
1.3	LUPR-F-01-03	Data yang harus direkam adalah berkas tugas, alamat <i>IP</i> , seluruh <i>window</i> , <i>window</i> yang aktif, nama <i>login</i> , dan nama piranti.		
1.4	LUPR-F-01-04	Sistem harus menampilkan notifikasi “ <i>Lup is recording</i> ” ketika perekaman mulai berjalan.		
1.5	LUPR-F-01-05	Sistem harus menyediakan tombol “ <i>Stop and Quit</i> ” sebagai sarana untuk menghentikan perekaman tugas dan keluar dari sistem.		
2	LUPR-F-02	Sistem harus menyediakan fungsi untuk mengubah nilai interval rekaman.	Mahasiswa	Mengubah interval rekaman.

No	Kode Kebutuhan	Deskripsi Kebutuhan	Aktor	Nama Use Case
2.1	LUPR-F-02-01	Sistem harus menyediakan tombol “ <i>Set Interval</i> ” sebagai sarana untuk mengubah interval rekaman.		
2.2	LUPR-F-02-02	Sistem harus menyediakan kolom dialog untuk mengisi nilai interval.		
2.3	LUPR-F-02-03	Nilai interval hanya dapat diisi angka bilangan bulat diatas 0.		
2.4	LUPR-F-02-04	Sistem harus menampilkan peringatan “ <i>Interval must higher than 0</i> ” jika kolom interval diisi dengan selain bilangan bulat diatas 0.		
2.5	LUPR-F-02-05	Sistem harus menonaktifkan tombol “ <i>Set Interval</i> ” ketika perekaman belum dijalankan.		
2.6	LUPR-F-02-06	Sistem harus menampilkan notifikasi “ <i>Interval changed to «nilai interval»</i> ” ketika nilai interval berhasil dirubah.		

Tabel 4.8 Spesifikasi kebutuhan fungsional, definisi, dan pemetaan nama *use case* sistem *Lup Viewer*

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
1	LUPV-F-01	Sistem harus menyediakan fungsi untuk menampilkan seluruh daftar rekaman mahasiswa.	Dosen	Melihat se-luruh daftar rekaman.
1.1	LUPV-F-01-01	Sistem harus menyediakan tombol “Open Records” sebagai sarana untuk memilih rekaman yang akan ditam-pulkan.		
1.2	LUPV-F-01-02	Sistem hanya dapat membaca direktori dengan for-mat “«namamahasiswa-nimmahasiswa»” (“alfabet-angka”).		
1.3	LUPV-F-01-03	Sistem hanya dapat membaca direktori yang memuat rekaman hasil dari sistem <i>Lup Recorder</i> .		
1.4	LUPV-F-01-04	Sistem harus menampilkan data berupa nama maha-siswa, NIM, total rekaman, waktu rekaman awal,waktu rekaman akhir, dan durasi penggerjaan tugas (<i>name, stu-dent ID, total records, first record, last record, work du-ration</i>).		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
1.5	LUPV-F-01-05	Sistem harus menampilkan pesan peringatan “ <i>Not a valid Task directory (baris baru) Contains invalid Task: «direktori yang tidak valid»</i> jika terdapat direktori yang tidak valid.		
2	LUPV-F-02	Sistem harus menyediakan fungsi untuk menampilkan hasil rekaman penggeraan tugas.	Dosen	Melihat hasil rekaman.
2.1	LUPV-F-02-01	Sistem harus memiliki tombol <i>combo box “Filename”</i> untuk memilih berkas tugas yang akan ditampilkan.		
2.2	LUPV-F-02-02	Sistem harus memiliki tombol <i>radio button “Diff Mode”</i> dan <i>“Show Mode”</i> sebagai sarana untuk mengalihkan mode tampilan rekaman berkas.		
2.3	LUPV-F-02-03	Sistem harus memiliki tombol <i>check box “Line Insertion and Deletion”</i> sebagai sarana untuk menampilkan jumlah baris yang dihapus dan ditambah pada berkas.		
2.4	LUPV-F-02-04	Sistem harus menampilkan seluruh daftar (per interval waktu) rekaman penggeraan tugas.		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
2.5	LUPV-F-02-05	Sistem harus menampilkan data berupa tanggal dan waktu rekaman, isi perubahan berkas, jumlah baris yang dihapus pada berkas, jumlah baris yang ditambah pada berkas, alamat <i>IP</i> , seluruh <i>window, window</i> yang aktif, nama <i>login</i> , dan nama piranti.		
2.6	LUPV-F-02-06	Sistem menampilkan pesan informasi “No file selected, please select one.” jika tidak ada berkas tugas yang dipilih.		
2.7	LUPV-F-02-07	Sistem menampilkan pesan informasi “No available record for «nama berkas» in this period” jika tidak ada perubahan tugas pada rekaman tertentu.		
2.8	LUPV-F-02-08	Sistem harus memiliki fungsi untuk mengalihkan isi perubahan berkas dari mode <i>diff</i> ke <i>show</i> mode dan sebaliknya.		
2.9	LUPV-F-02-09	Sistem harus memiliki fungsi untuk memberikan warna merah pada baris yang dihapus dan warna hijau pada baris yang ditambah dalam mode <i>diff</i> .		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
2.10	LUPV-F-02-10	Sistem harus menampilkan pesan peringatan “ <i>Please choose a file</i> ” jika tombol <i>check box</i> “ <i>Line Insertion and Deletion</i> ” diaktifkan tanpa ada berkas yang dipilih.		
3	LUPV-F-03	Sistem harus menyediakan fungsi untuk mencari tersangka plagiarisme.	Dosen	Mencari tersangka.
3.1	LUPV-F-03-01	Sistem harus menyediakan kolom “ <i>Line Insertion Limit</i> ” untuk mengisi nilai <i>threshold</i> atau <i>limit</i> pada baris yang ditambahkan di dalam rekaman berkas tugas.		
3.2	LUPV-F-03-02	Sistem harus menyediakan tombol <i>combo box</i> “ <i>Filename</i> ” sebagai sarana untuk memilih berkas tugas.		
3.3	LUPV-F-03-03	Sistem harus menyediakan tombol “ <i>Search</i> ” sebagai sarana untuk memulai proses pencarian.		
3.4	LUPV-F-03-04	Data yang ditampilkan adalah nama, nim, nama berkas, <i>line insertion</i> , dan waktu (<i>name, student ID, filename, line insertion, date time</i>).		
3.5	LUPV-F-03-05	Sistem harus menampilkan peringatan “ <i>Please select a file</i> ” jika tidak ada berkas yang dipilih.		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
3.6	LUPV-F-03-06	Sistem harus menampilkan peringatan " <i>Please set limit higher than 0 (baris baru) Above 10 is recommended</i> " jika kolom <i>line insertion limit</i> bernilai 0.		
3.7	LUPV-F-03-07	Sistem harus menampilkan informasi " <i>No suspect found, for «limit» insertion limit</i> " jika tidak ada tersangka yang ditemukan.		
4	LUPV-F-04	Sistem harus menyediakan fungsi untuk mencari window yang terbuka selama pengerjaan tugas.	Dosen	Mencari window.
4.1	LUPV-F-04-01	Sistem harus menyediakan kolom " <i>Window Name</i> " sebagai sarana untuk mengisi nama <i>window</i> yang akan dicari.		
4.2	LUPV-F-04-02	Sistem harus menyediakan tombol " <i>Search</i> " sebagai sarana untuk memulai proses pencarian.		
4.3	LUPV-F-04-03	Data yang ditampilkan adalah nama <i>window</i> , nama mahasiswa, nim, dan waktu (<i>window name, name, student ID, date time</i>).		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
4.4	LUPV-F-04-04	Sistem harus menampilkan pesan peringatan “ <i>Please supply the window name</i> ” apabila kolom nama <i>window</i> kosong.		
4.5	LUPV-F-04-05	Sistem harus menampilkan informasi “ <i>No window name for «input» found</i> ” jika tidak ada <i>window</i> yang ditemukan.		
5	LUPV-F-05	Sistem harus menyediakan fungsi untuk menampilkan alamat <i>IP</i> seluruh mahasiswa selama penggeraan tugas.	Dosen	Melihat seluruh alamat <i>IP</i> .
5.1	LUPV-F-05-01	Sistem harus menyediakan tombol “ <i>Show all IP addresses</i> ” sebagai sarana untuk menjalankan fungsi menampilkan seluruh alamat <i>IP</i> .		
5.2	LUPV-F-05-02	Data yang ditampilkan adalah alamat <i>IP</i> , nama mahasiswa, nim, dan waktu (<i>IP address, name, student ID, date time</i>).		
6	LUPV-F-06	Sistem harus menyediakan fungsi menampilkan grafik <i>edit-distance</i> .	Dosen	Melihat grafik <i>edit-distance</i> .
6.1	LUPV-F-06-01	Sistem harus menyediakan tombol “ <i>Show</i> ” sebagai sara-na untuk menampilkan grafik <i>edit-distance</i> .		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
6.2	LUPV-F-06-02	Sistem harus menyediakan tombol <i>combo box</i> “ <i>Current Student</i> ” sebagai sarana untuk memilih mahasiswa.		
6.3	LUPV-F-06-03	Sistem harus menyediakan tombol <i>combo box</i> “ <i>Filename</i> ” sebagai sarana untuk memilih nama berkas.		
6.4	LUPV-F-06-04	Sistem harus menampilkan pesan peringatan “ <i>Please fill one of the pairs</i> ” jika salah satu atau kedua <i>combo box</i> “ <i>Filename</i> ” dan “ <i>Current Student</i> ” tidak diisi.		
6.5	LUPV-F-06-05	Data yang harus ada dalam grafik <i>edit-distance</i> adalah nama mahasiswa, nim, legenda rekaman, dan legenda <i>edit-distance</i> .		
7	LUPV-F-07	Sistem harus menyediakan fungsi untuk melihat grafik dari nilai <i>edit-distance</i> tugas sebelumnya.	Dosen	Melihat grafik <i>edit-distance</i> tugas sebelumnya.
7.1	LUPV-F-07-01	Sistem harus menyediakan tombol “ <i>Load Edit-distance File</i> ” sebagai sarana untuk memilih berkas <i>edit-distance</i> sebelumnya.		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
7.2	LUPV-F-07-02	Sistem harus menampilkan pesan peringatan " <i>Please load edit-distance file first</i> " jika berkas sebelumnya belum dimuat.		
7.3	LUPV-F-07-03	Sistem harus menggunakan fungsionalitas yang telah ada pada spesifikasi kebutuhan LUPV-F-06-01 hingga LUPV-F-06-05, dengan modifikasi nama kolom " <i>Current Student</i> " menjadi " <i>Previous Student</i> "		
8	LUPV-F-08	Sistem harus menyediakan fungsi menyimpan grafik <i>edit-distance</i> .	Dosen	Menyimpan grafik <i>edit-distance</i> .
8.1	LUPV-F-08-01	Sistem harus menyediakan tombol " <i>Save Graph</i> " sebagai sarana untuk menyimpan berkas.		
8.2	LUPV-F-08-02	Sistem harus menyimpan grafik pada direktori tugas dengan membuat direktori baru bernama " <i>lupv-notes</i> ".		
8.3	LUPV-F-08-03	Sistem harus menyimpan grafik dengan format " <i>namamahasiswa-nimmahasiswa».png</i> ", dan menggunakan tanda hubung "_" jika terdapat dua mahasiswa.		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
8.4	LUPV-F-08-04	Sistem harus menggunakan fungsionalitas yang telah ada pada spesifikasi kebutuhan LUPV-F-06-04.		
8.5	LUPV-F-08-05	Sistem harus menampilkan pesan informasi “ <i>Graph saved to «lokasi berkas»</i> ” jika grafik berhasil disimpan.		
9	LUPV-F-09	Sistem harus menyediakan fungsi untuk mengekspor semua nilai <i>edit-distance</i> .	Dosen	Mengekspor semua nilai <i>edit-distance</i> .
9.1	LUPV-F-09-01	Data yang harus ada adalah nama mahasiswa, nim, perhitungan rekaman, nilai <i>edit-distance</i> , dan nama tugas.		
9.2	LUPV-F-09-02	Sistem harus menyediakan tombol “ <i>Export Edit-distance</i> ” sebagai sarana untuk mengekspor semua nilai <i>edit-distance</i> .		
9.3	LUPV-F-09-03	Sistem harus menyediakan <i>dialog window</i> yang di dalamnya terdapat tombol <i>combo box</i> untuk memilih berkas yang akan diekspor.		
9.4	LUPV-F-09-04	Sistem harus menyimpan berkas ekspor pada direktori tugas dengan membuat direktori baru bernama “ <i>lupv-notes</i> ”.		

No	Kode Kebutuhan	Definisi Kebutuhan	Aktor	Nama Use Case
9.5	LUPV-F-09-05	Sistem harus menyimpan berkas ekspor dengan nama berkas “«namatugas»-editdistance.lup”.		
9.6	LUPV-F-09-06	Sistem harus menampilkan pesan informasi “ <i>Edit-distance exported to «lokasi berkas»</i> ” jika ekspor berhasil.		
10	LUPV-F-10	Sistem harus menyediakan fungsi untuk mengalihkan format waktu.	Dosen	Mengalihkan format waktu.
10.1	LUPV-F-10-01	Sistem harus menyediakan dua format tanggal untuk dialihkan, yaitu format <i>real date time</i> dan format <i>relative date time</i> .		
10.2	LUPV-F-10-02	Format <i>real date time</i> berbentuk “«nama pendek hari, tanggal bulan tahun, jam:menit:detik»” dan format <i>relative date time</i> berbentuk “«x hours ago»”.		
10.3	LUPV-F-10-03	Sistem harus menyediakan tombol <i>radio button</i> “ <i>Relative DateTime</i> ” dan “ <i>Real DateTime</i> ” sebagai sarana untuk mengalihkan antar format waktu.		

4.5 Kebutuhan Non-Fungsional Sistem

Kebutuhan non-fungsional merupakan batasan (*constraint*) dari sebuah sistem (Sommerville, 2014). Daftar kebutuhan non-fungsional didapatkan dari proses analisis kebutuhan yang telah dilakukan sebelumnya. Dalam penelitian ini, kebutuhan non-fungsional memiliki sistem penomoran LUPV-NF-XX-YY untuk sistem *Lup Viewer* dan LUPR-NF-XX-YY untuk sistem *Lup Recorder*. *LUPV* dan *LUPR* merupakan nama sistem, NF menunjukkan tipe kebutuhan yaitu kebutuhan non-fungsional, XX menunjukkan nomor kebutuhan, dan YY menunjukkan nomor spesifikasi kebutuhan. Tabel 4.9 dan Tabel 4.10 memaparkan daftar kebutuhan non-fungsional, definisi, dan parameter kebutuhan non-fungsional.

Tabel 4.9 Spesifikasi kebutuhan non-fungsional, parameter, dan deskripsi kebutuhan sistem *Lup Recorder*

No	Kode Kebutuhan	Parameter	Deskripsi Kebutuhan
1	LUPR-NF-01	<i>compatibility</i>	Sistem harus berjalan dengan baik pada enam <i>desktop environment</i> yang berbeda.

Tabel 4.10 Spesifikasi kebutuhan non-fungsional, parameter, dan deskripsi kebutuhan sistem *Lup Viewer*

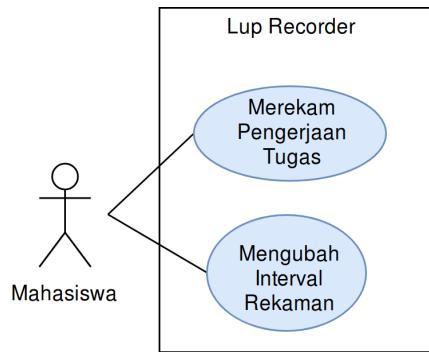
No	Kode Kebutuhan	Parameter	Deskripsi Kebutuhan
1	LUPV-NF-01	<i>compatibility</i>	Sistem harus berjalan dengan baik pada enam <i>desktop environment</i> yang berbeda.

4.6 Pemodelan Kebutuhan

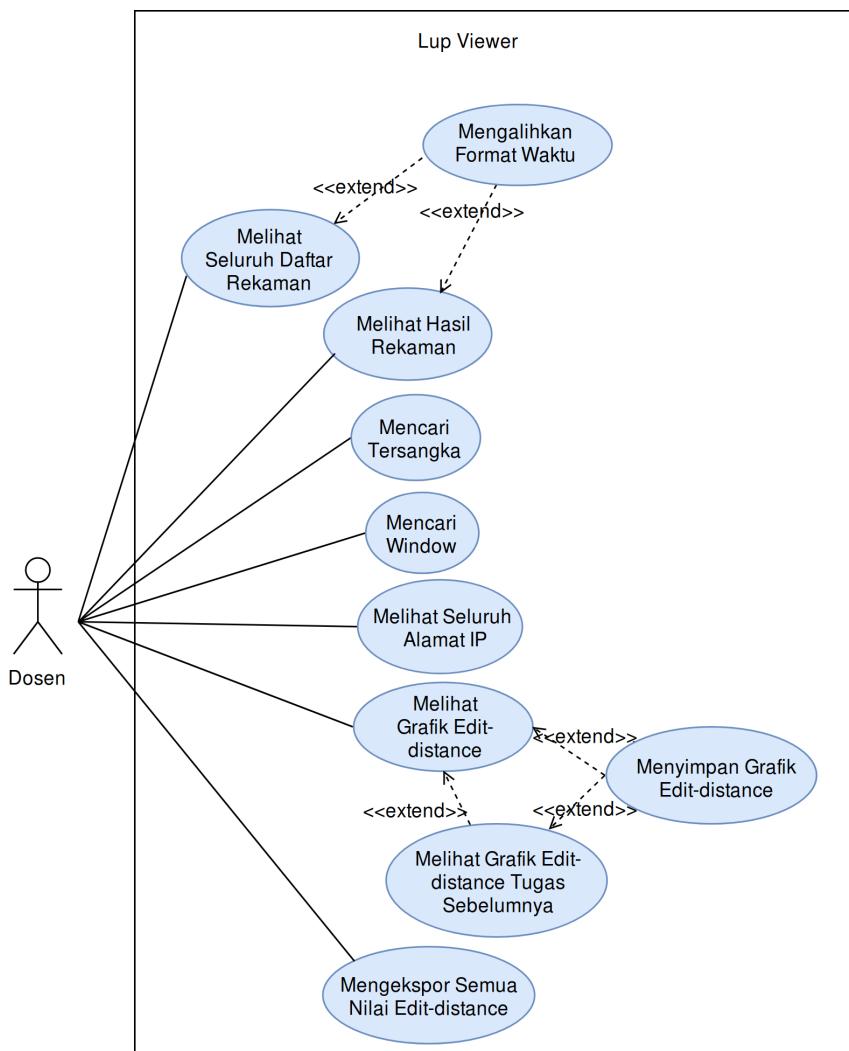
4.6.1 *Use Case Diagram*

Use case diagram digunakan untuk menggambarkan fungsionalitas dan batasan sistem secara visual. Selain itu, *use case diagram* juga menggambarkan interaksi antar aktor dengan sistem dari sudut aktor dalam bentuk visual. Aktor ditun-

rukkan dengan *stick figure* dan *use case* ditunjukkan dengan oval. *Use case* yang digambarkan didapatkan dari hasil analisis kebutuhan sebelumnya. Gambar 4.15 menunjukkan *use case diagram* untuk sistem *Lup Recorder* dan Gambar 4.16 menunjukkan *use case diagram* untuk sistem *Lup Viewer*.



Gambar 4.15 Use case diagram Lup Recorder



Gambar 4.16 Use case diagram Lup Viewer

4.6.2 Use Case Scenario

Use case scenario digunakan untuk menjelaskan skenario interaksi antar aktor dan sistem secara tekstual dengan detail. Dalam setiap *use case scenario* dipaparkan *objective* atau tujuan *use case*, aktor yang terlibat, kondisi sebelumnya yang harus dipenuhi *pre-condition*, alur utama *main flow*, alur alternatif *alternative flow*, dan kondisi setelah *use case* dijalankan *post-condition*. *Use case scenario* untuk sistem *Lup Recorder* dipaparkan pada Tabel 4.11 hingga Tabel 4.12, dan *use case scenario* sistem *Lup Viewer* dipaparkan pada Tabel 4.13 hingga 4.22.

Tabel 4.11 Use case scenario untuk Merekam Pengerajan Tugas

<i>Flow of events</i> untuk Merekam Pengerajan Tugas	
Kode Kebutuhan	LUPR-F-01
Objective	Mahasiswa dapat merekam pengerajan tugas.
Actor	Mahasiswa
Pre-condition	Tampilan utama sistem <i>Lup Recorder</i> sudah tersedia.
Main flow	<ol style="list-style-type: none">1. Mahasiswa memilih tombol “Start Recording” untuk mulai merekam pengerajan tugas.2. Sistem menampilkan dialog pemilihan letak direktori tugas. Pada dialog tersedia tombol “Choose” dan “Cancel”.3. Mahasiswa memilih letak direktori tugas dan memilih tombol “Choose”.4. Sistem menjalankan perekaman pengerajan tugas dengan data yang direkam meliputi berkas tugas, alamat IP, seluruh window, window yang aktif, nama login, dan nama piranti.5. Sistem menampilkan pesan notifikasi “<i>Lup is recording</i>” menandakan bahwa perekaman sudah dijalankan.
Alternative flows	<ol style="list-style-type: none">1.1 Sistem berhenti dan keluar jika mahasiswa memilih tombol “Stop and Quit”.2.1 Sistem menutup dialog jika mahasiswa memilih tombol “Cancel”.
Post-condition	Pengerjaan tugas direkam oleh sistem.

Tabel 4.12 Use case scenario untuk Mengubah Interval Rekaman

<i>Flow of events</i> untuk Mengubah Interval Rekaman	
Kode Kebutuhan	LUPR-F-02
Objective	Mahasiswa dapat mengubah interval rekaman.
Actor	Mahasiswa
Pre-condition	Perekaman tugas sudah dijalankan.
Main flow	<ol style="list-style-type: none"> 1. Mahasiswa memilih tombol “Set Interval”. 2. Sistem menampilkan dialog interval yang terdiri dari kolom nilai interval. Pada dialog interval tersedia tombol “Apply” dan “Cancel”. 3. Mahasiswa memberikan nilai bilangan bulat diatas 0 pada kolom interval. 4. Mahasiswa memilih tombol “Apply” pada kolom interval. 5. Sistem melakukan validasi nilai interval dan mengubah nilai interval rekaman. 6. Sistem menampilkan pesan notifikasi “<i>Interval changed to «nilai interval»</i>” menandakan bahwa nilai interval berhasil dirubah.
Alternative flows	<p>5.1 Apabila kolom bernilai selain bilangan bulat diatas 0, maka sistem menampilkan pesan peringatan “<i>Interval must higher than 0</i>”.</p>
Post-condition	Nilai interval rekaman diubah oleh sistem.

Tabel 4.13 Use case scenario untuk Melihat Seluruh Daftar Rekaman

<i>Flow of events</i> untuk Melihat Seluruh Daftar Rekaman	
Kode Kebutuhan	LUPV-F-01
Objective	Dosen dapat melihat seluruh daftar rekaman.
Actor	Dosen
Pre-condition	Tampilan utama sistem <i>Lup Viewer</i> sudah tersedia.

Main flow	<ol style="list-style-type: none"> 1. Dosen memilih tombol “Open Records”. 2. Sistem menampilkan dialog untuk memilih direktori rekaman yang akan dibuka. Pada dialog tersebut terdapat tombol “Choose” dan “Cancel”. 3. Dosen memilih tombol “Choose”. 4. Sistem melakukan validasi direktori rekaman dan menampilkan seluruh daftar rekaman mahasiswa. Data yang ditampilkan meliputi nama, nim, total rekaman, waktu rekaman awal, waktu rekaman akhir, dan durasi pengerajan.
Alternative flows	<ol style="list-style-type: none"> 3.1 Apabila dosen memilih tombol “Cancel”, maka berkas rekaman tidak dibuka. 4.1 Apabila direktori tidak valid, yaitu tidak sesuai dengan spesifikasi kebutuhan LUPV-F-01-02 dan LUPV-F-01-03, maka sistem menampilkan pesan peringatan “<i>Not a valid Task directory (baris baru)</i> Contains invalid Task: «direktori yang tidak valid»”. 4.2 Perluasan ke <i>use case</i> dengan kode kebutuhan LUPV-F-10.
Post-condition	Seluruh daftar rekaman ditampilkan oleh sistem.

Tabel 4.14 Use case scenario untuk Menampilkan Hasil Rekaman

<i>Flow of events</i> untuk Melihat Hasil Rekaman	
Kode Kebutuhan	LUPV-F-02
Objective	Dosen dapat melihat hasil rekaman.
Actor	Dosen
Pre-condition	Tampilan seluruh daftar rekaman mahasiswa sudah tersedia.

Main flow	<ol style="list-style-type: none"> 1. Dosen memilih <i>clickable</i> baris mahasiswa pada seluruh daftar rekaman. 2. Sistem menampilkan <i>clickable list</i> daftar rekaman. 3. Dosen memilih <i>clickable list</i> daftar rekaman. 4. Sistem menampilkan rekaman alamat <i>IP</i>, seluruh <i>window</i>, <i>window</i> yang aktif, nama <i>login</i>, dan nama <i>piranti</i>. 5. Dosen memilih nama berkas pada tombol <i>combo box</i> “<i>Filename</i>”. 6. Sistem menampilkan hasil rekaman isi perubahan berkas. 7. Dosen memilih tombol “<i>Line Insertion and Deletion</i>”. 8. Sistem menampilkan jumlah baris yang dihapus dan jumlah baris yang ditambah pada berkas. 9. Dosen memilih tombol “<i>Show Mode</i>” dan “<i>Diff Mode</i>”. 10. Sistem menampilkan format perubahan berkas dengan mode <i>show</i> dan mode <i>diff</i>.
Alternative flows	<p>4.1 Perluasan ke <i>use case</i> dengan kode kebutuhan LUPV-F-10.</p> <p>6.1 Sistem menampilkan pesan informasi “<i>No file selected, please select one</i>” jika tidak ada berkas tugas yang dipilih.</p> <p>6.2 Sistem menampilkan pesan informasi “<i>No available record for «nama berkas» in this period</i>” jika tidak ada perubahan tugas pada rekaman yang dipilih.</p> <p>7.1 Sistem menampilkan pesan peringatan “<i>Please choose a file</i>” jika tidak ada berkas yang dipilih.</p>
Post-condition	Hasil rekaman ditampilkan oleh sistem.

Tabel 4.15 Use case scenario untuk Mencari Tersangka.

<i>Flow of events</i> untuk Mencari Tersangka	
Kode Kebutuhan	LUPV-F-03
Objective	Dosen dapat mencari tersangka.

Actor	Dosen
Pre-condition	Tampilan pencarian tersangka sudah tersedia.
Main flow	<ol style="list-style-type: none"> 1. Dosen memilih nama berkas dari tombol <i>combo box “Filename”</i> dan mengisi nilai <i>“Line Insertion Limit”</i>. 2. Dosen memilih tombol <i>“Search”</i> untuk menjalankan pencarian. 3. Sistem melakukan validasi nilai <i>“Line Insertion Limit”</i> dan menampilkan hasil pencarian tersangka.
Alternative flows	<ol style="list-style-type: none"> 3.1 Sistem menampilkan pesan peringatan <i>“Please set limit higher than 0 (baris baru) Above 10 is recommended”</i> apabila nilai pada kolom <i>insertion</i> bernilai selain bilangan bulat di atas 0. 3.2 Sistem menampilkan pesan informasi <i>“No suspect found, for «limit» insertion limit”</i> jika tidak ada tersangka yang ditemukan. 3.3 Sistem menampilkan pesan peringatan <i>“Please select a file”</i> apabila tidak ada berkas tugas yang dipilih.
Post-condition	Hasil pencarian tersangka ditampilkan oleh sistem.

Tabel 4.16 Use case scenario untuk Mencari Window

<i>Flow of events</i> untuk Mencari Window	
Kode Kebutuhan	LUPV-F-04
Objective	Dosen dapat mencari window.
Actor	Dosen
Pre-condition	Tampilan pencarian window telah tersedia.
Main flow	<ol style="list-style-type: none"> 1. Dosen mengisi kolom <i>“Window Name”</i>. 2. Dosen memilih tombol <i>“Search”</i> untuk menjalankan pencarian. 3. Sistem melakukan validasi nilai kolom <i>“Window Name”</i> dan menampilkan hasil pencarian window.

Alternative flows	<p>3.1 Sistem menampilkan pesan informasi “<i>No window name for «nama window» found</i>” jika tidak ada <i>window</i> yang ditemukan.</p> <p>3.1 Sistem menampilkan pesan peringatan “<i>Please supply the window name</i>” apabila kolom “<i>Window Name</i>” kosong.</p>
Post-condition	Hasil pencarian <i>window</i> ditampilkan oleh sistem.

Tabel 4.17 Use case scenario untuk Menampilkan Seluruh Alamat IP

<i>Flow of events</i> untuk Melihat Seluruh Alamat IP	
Kode Kebutuhan	LUPV-F-05
Objective	Dosen dapat melihat seluruh alamat IP.
Actor	Dosen
Pre-condition	Tampilan untuk melihat seluruh alamat IP telah tersedia.
Main flow	<ol style="list-style-type: none"> 1. Dosen memilih tombol “<i>Show all IP addresses</i>”. 2. Sistem menampilkan seluruh alamat IP.
Alternative flows	-
Post-condition	Seluruh alamat IP ditampilkan oleh sistem

Tabel 4.18 Use case scenario untuk Melihat Grafik Edit-distance.

<i>Flow of events</i> untuk Melihat Grafik Edit-distance	
Kode Kebutuhan	LUPV-F-06
Objective	Dosen dapat melihat grafik edit-distance.
Actor	Dosen
Pre-condition	Tampilan untuk melihat grafik edit-distance telah tersedia.
Main flow	<ol style="list-style-type: none"> 1. Dosen memilih mahasiswa dari tombol <i>combo box</i> “<i>Current Student</i>” dan memilih berkas dari tombol <i>combo box</i> “<i>Filenamne</i>”. 2. Dosen memilih tombol “<i>Show Mode</i>”. 3. Sistem melakukan validasi nilai tombol <i>combo box</i> “<i>Current Student</i>” dan “<i>Filenamne</i>”, kemudian menampilkan grafik edit distance.

Alternative flows	1.2 Perluasan ke <i>use case</i> dengan kode kebutuhan LUPV-F-07 dan LUPV-F-08. 2.1 Sistem menampilkan pesan peringatan “ <i>Please fill one of the pairs</i> ” jika salah satu atau kedua <i>combo box</i> “ <i>Filename</i> ” dan “ <i>Current Student</i> ” tidak diisi.
Post-condition	Grafik <i>edit distance</i> ditampilkan oleh sistem.

Tabel 4.19 Use case scenario untuk Melihat Grafik *Edit-distance* Tugas Sebelumnya

<i>Flow of events</i> untuk Melihat Grafik <i>Edit-distance</i> Tugas Sebelumnya	
Kode Kebutuhan	LUPV-F-07
Objective	Dosen dapat melihat grafik <i>edit-distance</i> tugas sebelumnya.
Actor	Dosen
Pre-condition	Tampilan untuk melihat grafik <i>edit-distance</i> telah tersedia.
Main flow	<ol style="list-style-type: none"> Dosen memilih tombol “<i>Load Edit-distance</i>”. Sistem menampilkan dialog untuk memilih berkas <i>edit-distance</i> tugas sebelumnya. Dialog memiliki tombol “<i>Open</i>” dan “<i>Cancel</i>”. Dosen memilih berkas dan memilih tombol “<i>Open</i>”. Sistem membaca berkas <i>edit-distance</i> tugas sebelumnya. Dosen memilih mahasiswa dari tombol <i>combo box</i> “<i>Previous Student</i>” dan memilih berkas dari tombol <i>combo box</i> “<i>Filename</i>”. Dosen memilih tombol “<i>Show Mode</i>”. Sistem melakukan validasi nilai tombol <i>combo box</i> “<i>Previous Student</i>” dan “<i>Filename</i>”, kemudian menampilkan grafik <i>edit distance</i> tugas sebelumnya.

Alternative flows	<p>3.1 Sistem tidak membaca berkas jika dosen memilih tombol “Cancel”.</p> <p>5.1 Sistem menampilkan pesan peringatan “Please load edit-distance file first” jika dosen belum membuka berkas <i>edit-distance</i> tugas sebelumnya.</p> <p>5.2 Sistem menampilkan pesan peringatan “Please fill one of the pairs” jika salah satu atau kedua <i>combo box</i> “Filename” dan “Previous Student” belum dipilih.</p> <p>6.1 Perluasan ke <i>use case</i> dengan kode kebutuhan LUPV-F-08.</p>
Post-condition	Grafik <i>edit distance</i> tugas sebelumnya ditampilkan oleh sistem.

Tabel 4.20 Use case scenario untuk Menyimpan Grafik Edit-distance .

<i>Flow of events</i> untuk Menyimpan Grafik <i>Edit-distance</i>	
Kode Kebutuhan	LUPV-F-08
Objective	Dosen dapat menyimpan grafik <i>edit-distance</i> .
Actor	Dosen
Pre-condition	Grafik <i>edit-distance</i> telah tersedia.
Main flow	<ol style="list-style-type: none"> 1. Dosen memilih tombol “Save Graph”. 2. Sistem menyimpan grafik pada direktori “lupv-notes” pada direktori tugas. 3. Sistem menampilkan pesan informasi “Graph saved to «lokasi berkas»” menandakan bahwa grafik telah disimpan.
Alternative flows	-
Post-condition	Grafik <i>edit distance</i> disimpan oleh sistem.

Tabel 4.21 Use case scenario untuk Mengekspor Semua Nilai Edit-distance

<i>Flow of events</i> untuk Mengekspor Semua Nilai <i>Edit-distance</i>	
Kode Kebutuhan	LUPV-F-09
Objective	Dosen dapat mengekspor semua nilai <i>edit-distance</i> .
Actor	Dosen

Pre-condition	Tampilan seluruh daftar rekaman mahasiswa telah tersedia.
Main flow	<ol style="list-style-type: none"> 1. Dosen memilih tombol “<i>Export Edit-distance</i>”. 2. Sistem menampilkan dialog pemilihan berkas yang akan di ekspor. Pada dialog tersedia tombol “<i>Export</i>” dan “<i>Cancel</i>”. 3. Dosen memilih tombol “<i>Export</i>”. 4. Sistem mengekspor semua nilai <i>edit-distance</i> pada berkas yang dipilih dan menyimpan hasil ekspor dengan nama “«namatugas»-editdistance.lup”. 5. Sistem menampilkan pesan informasi “<i>Edit-distance exported to «lokasi berkas»</i>” menandakan bahwa semua nilai <i>edit-distance</i> telah diekspor.
Alternative flows	<p>3.1 Apabila dosen memilih tombol “<i>Cancel</i>”, maka sistem menutup dialog pemilihan berkas dan nilai <i>edit-distance</i> tidak diekspor.</p>
Post-condition	Semua nilai <i>edit-distance</i> diekspor oleh sistem.

Tabel 4.22 Use case scenario untuk Mengalihkan Format Waktu

Flow of events untuk Mengalihkan Format Waktu	
Kode Kebutuhan	LUPV-F-10
Objective	Dosen dapat mengalihkan format waktu.
Actor	Dosen
Pre-condition	Tampilan waktu rekaman telah tersedia.
Main flow	<ol style="list-style-type: none"> 1. Dosen memilih tombol “<i>Real DateTime</i>” dan “<i>Relative DateTime</i>”. 2. Sistem mengalihkan format waktu. Format bentuk “«nama pendek hari, tanggal bulan tahun, jam:menit:detik»” untuk format <i>real</i> dan “«x hours ago»” untuk format <i>relative</i>.
Alternative flows	-
Post-condition	Format waktu dialihkan oleh sistem.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan Sistem

Tahapan perancangan sistem merupakan sebuah tahapan yang dilakukan setelah selesainya proses rekayasa kebutuhan. Hasil tahapan perancangan nantinya digunakan sebagai landasan tahapan implementasi sistem. Tiga bagian yang terdapat pada tahapan ini adalah perancangan arsitektur, perancangan komponen, dan perancangan antarmuka.

5.1.1 Perancangan Arsitektur

Sistem dibangun menggunakan arsitektur *Model-View-Controller (MVC)*. Maka komponen fungsionalitas inti sistem terpisah dari komponen antarmuka sistem. Pemisahan ini memberikan manfaat untuk pengembangan sistem pada tahap selanjutnya. Salah satu manfaat tersebut adalah kemudahan dalam mengembangkan antarmuka lain untuk sistem, hal ini dikarenakan pengembang hanya fokus pada komponen antarmuka yang akan dibangun. Perancangan arsitektur dilakukan dengan menggunakan notasi *UML sequence diagram* dan *class diagram*. *Class diagram* bertujuan menggambarkan struktur sistem dan relasi antar komponen dalam sistem secara statis. Pada *class diagram* digambarkan *class*, *method* atau *function*, dan relasi antar *class*. *Sequence diagram* merupakan diagram dinamis yang berfungsi untuk menggambarkan interaksi antar komponen atau objek yang berada di dalam sistem. Terdapat tiga sampel *sequence diagram* yang akan dipaparkan pada bagian ini. Tiga sampel tersebut adalah *sequence diagram* untuk merekam pengerjaan tugas, *sequence diagram* untuk melihat hasil rekaman, dan *sequence diagram* untuk melihat grafik *edit-distance* tugas sebelumnya. Sementara itu, *class diagram* sistem akan dipaparkan seluruhnya.

5.1.1.1 *Sequence Diagram* Merekam Pengerjaan Tugas

Sequence diagram untuk merekam pengerjaan tugas dapat dilihat pada Gambar 5.17. Pada *sequence diagram* tersebut terdapat beberapa objek yang berinteraksi. Satu objek *boundary* “*MainView*”, dua objek *controller* “*Controller*” dan “*RecordWorker*”, dan satu objek *entity* “*Model*”. Aktor yang terlibat adalah mahasiswa.

5.1.1.2 Sequence Diagram Melihat Hasil Rekaman

Sequence diagram untuk melihat hasil rekaman dapat dilihat pada Gambar 5.18. Pada *sequence diagram* tersebut terdapat beberapa objek yang berinteraksi. Dua objek *boundary* “MainView” dan “LogView”, dua objek *controller* “LogController” dan “MainController”, dan satu objek *entity* “LogModel”. Aktor yang terlibat adalah dosen.

5.1.1.3 Sequence Diagram Melihat Grafik Edit-Distance Tugas Sebelumnya

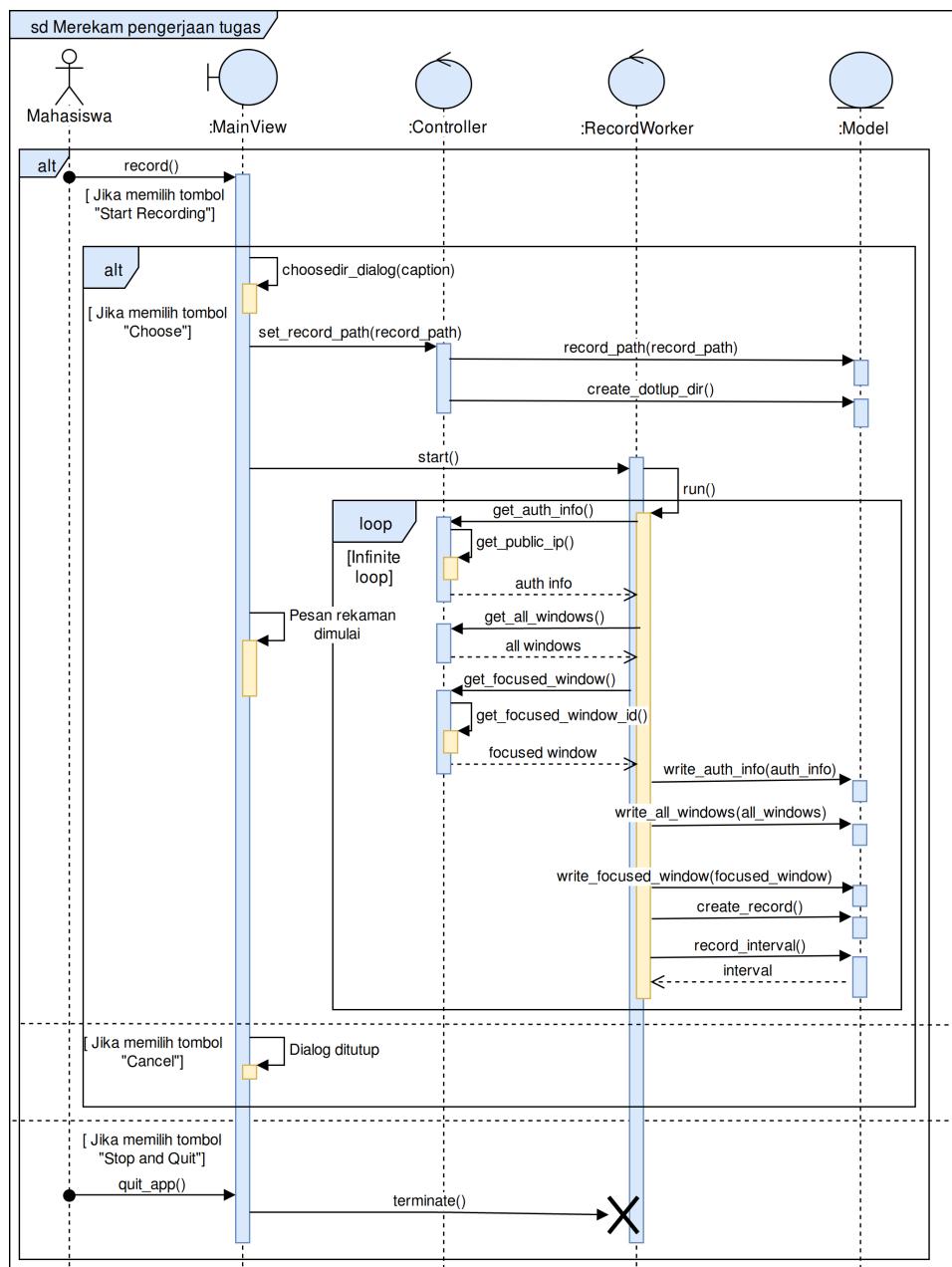
Sequence diagram untuk melihat grafik *edit-distance* tugas sebelumnya dapat dilihat pada Gambar 5.19. Pada *sequence diagram* tersebut terdapat beberapa objek yang berinteraksi. Satu objek *boundary* “SearchView”, satu objek *controller* “SearchController”, dan dua objek *entity* “searchModel” dan “MainModel”. Aktor yang terlibat adalah dosen.

5.1.1.4 Class Diagram

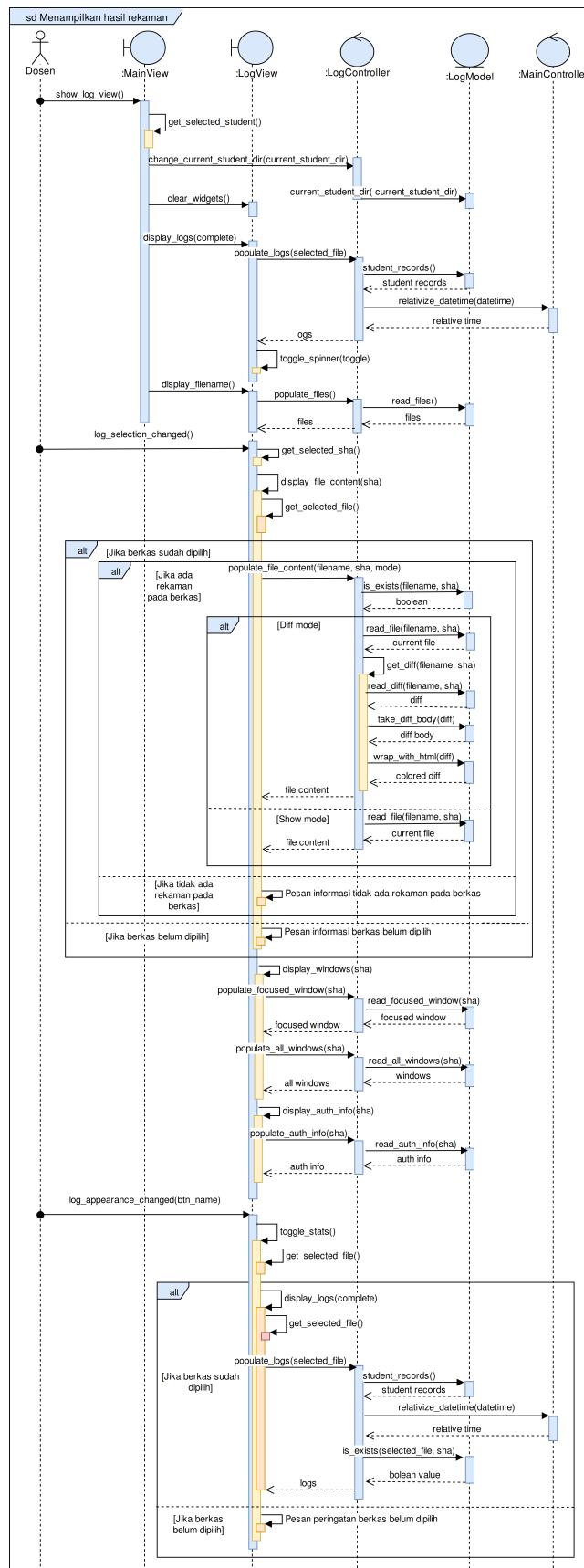
Pada bagian ini akan dipaparkan *class-class* yang menyusun sistem yang dibangun. Gambar 5.20 merupakan *class diagram* dari sistem *Lup Recorder*, dan Gambar 5.21 merupakan *class diagram* dari sistem *Lup Viewer*. Sistem yang dibangun menggunakan arsitektur *Model-View-Controller* sehingga *class-class* yang menyusun sistem pada *class diagram* umumnya terdiri dari *class Model*, *class View* dan *class Controller*. Pada *class diagram* juga dipaparkan relasi antar *class* yang umumnya merupakan relasi agregasi dan komposisi.

5.1.2 Perancangan Komponen

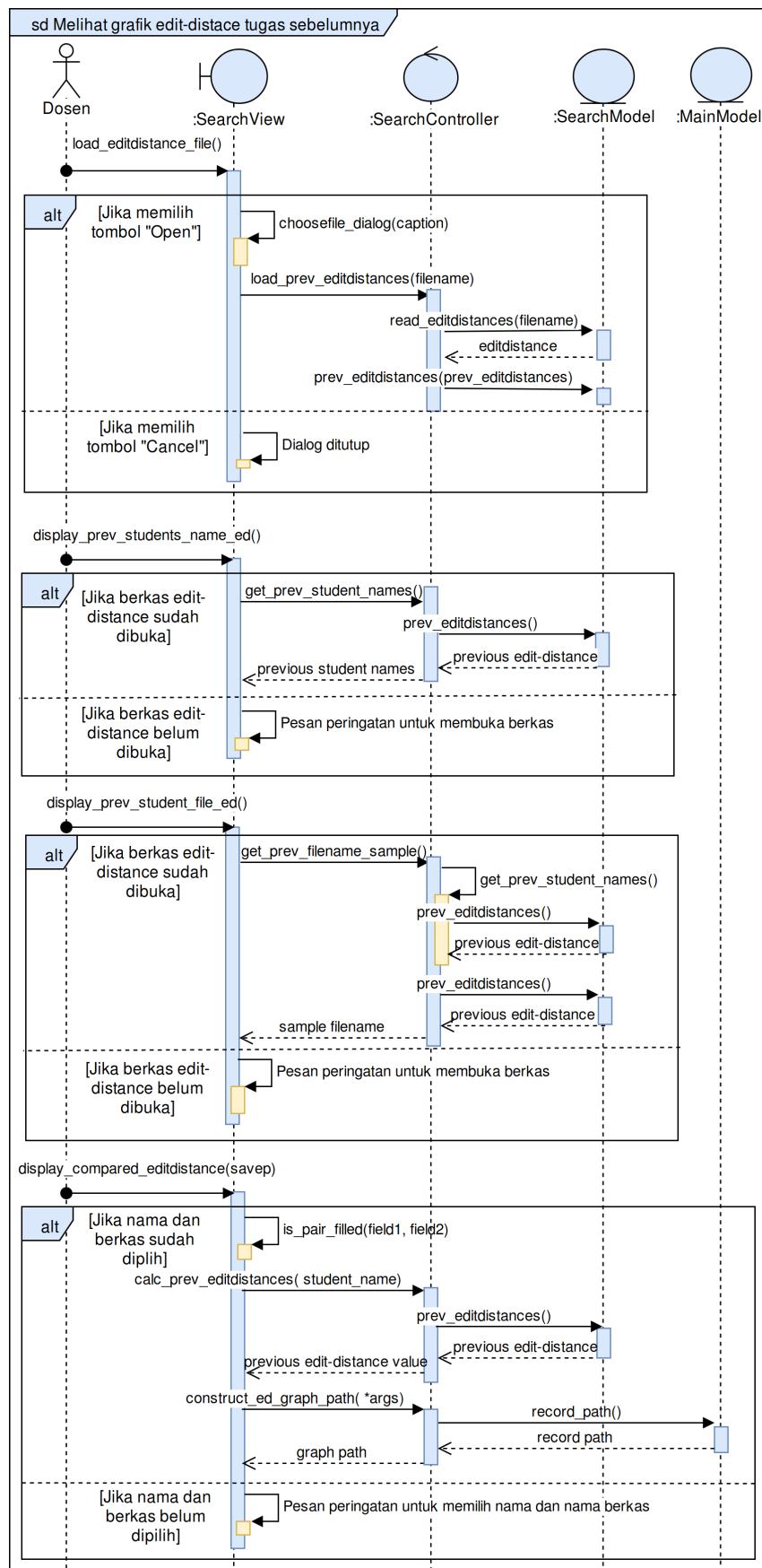
Pada tahapan perancangan komponen, komponen-komponen yang membentuk sistem akan dirancang menggunakan *pseudocode*. Komponen yang membentuk sistem dalam sistem berorientasi objek adalah *class*. Maka tahapan ini akan memaparkan perancangan tiga sampel *class*. Tidak semua *procedure* atau *function* di dalam suatu *class* akan dipaparkan, melainkan hanya satu sampel *function* dari tiap-tiap *class*. Tabel Kode 5.5 memaparkan *pseudocode* untuk *function get_all_windows* dari *class Controller* pada sistem *Lup Recorder*. Tabel Kode 5.6 memaparkan *pseudocode* untuk *function populate_logs* dari *class LogController* pada sistem *Lup Viewer* dan, Tabel Kode 5.7 memaparkan *pseudocode* untuk *function construct_ed_graph_path* dari *class SearchController* pada sistem *Lup Viewer*.



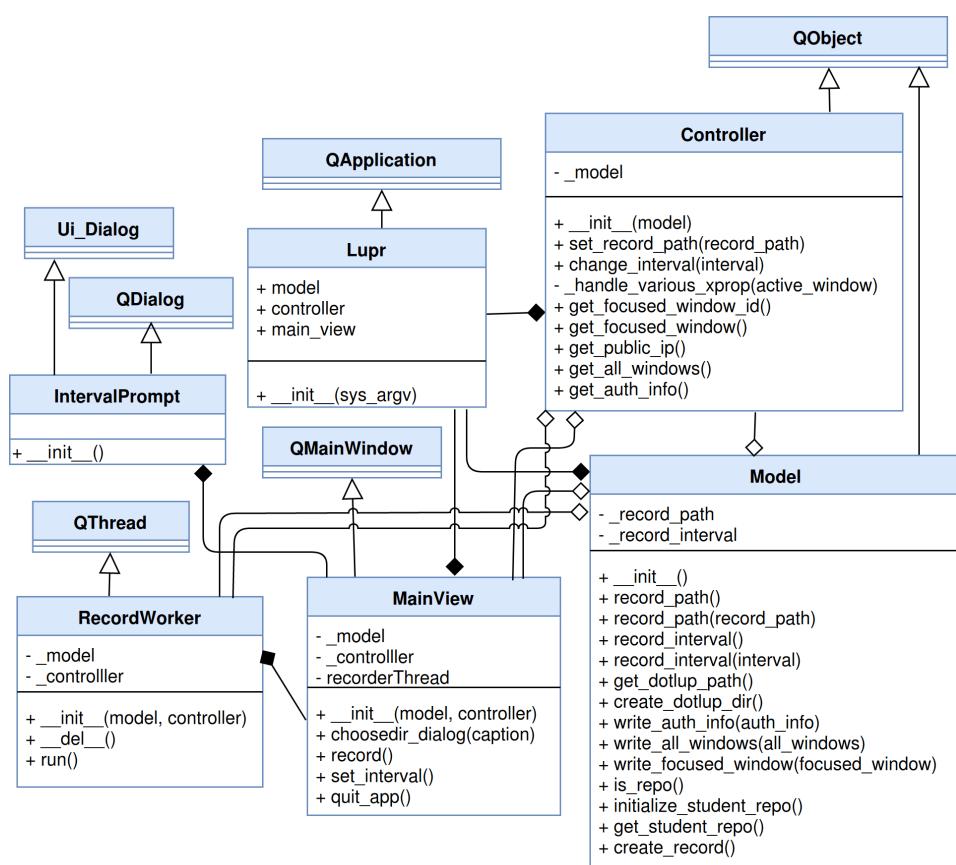
Gambar 5.17 Sequence diagram Merekam Pengerjaan Tugas



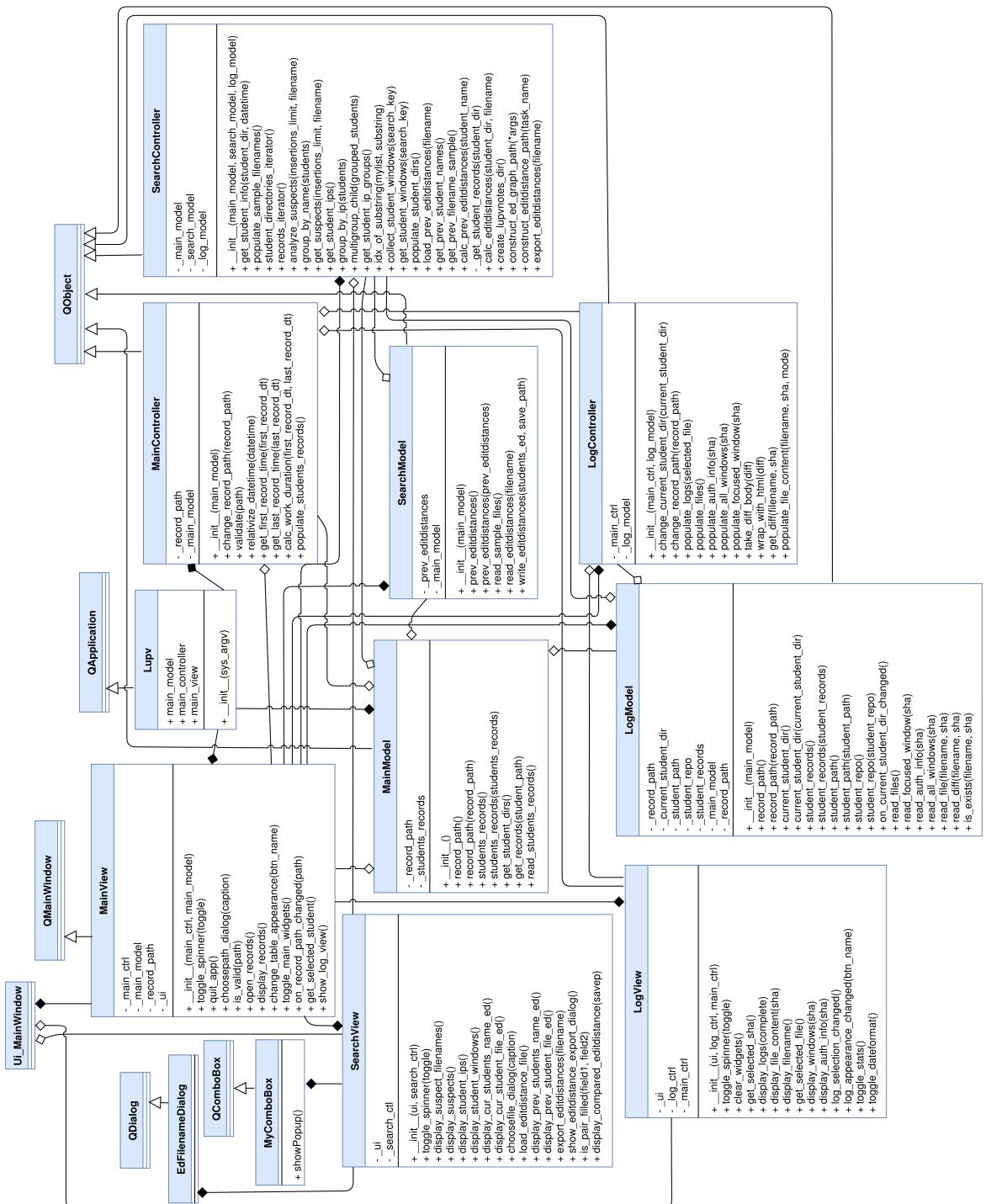
Gambar 5.18 Sequence diagram Melihat Hasil Rekaman



Gambar 5.19 *Sequence diagram Melihat Grafik Edit-Distance Tugas Sebelumnya*



Gambar 5.20 Pemodelan class diagram Lup Recorder



Gambar 5.21 Pemodelan *class diagram Lup Viewer*

5.1.2.1 Perancangan Komponen *Class Controller*

Pada Tabel Kode 5.5 dipaparkan *pseudocode get_all_windows* pada sistem *Lup Recorder*. *Pseudocode get_all_windows* dirancang untuk membaca seluruh *window* yang sedang dibuka. Sistem operasi memiliki banyak atribut selain nama *window*, maka *get_all_windows* hanya mengambil nilai nama *window* dan mengembalikannya.

No	get_all_windows
1	START 2 READ all_windows_dirty 3 IF all_windows_dirty is true 4 SET all_windows_dirty to split every line 5 FOR each window in all_windows_dirty 6 GET window_name 7 SET all_windows = window_name 8 ENDFOR 9 ENDIF 10 RETURN all_windows 11 END

Tabel Kode 5.5 Pseudocode function *get_all_windows*

5.1.2.2 Perancangan Komponen *Class LogController*

Pada Tabel Kode 5.6 dipaparkan *pseudocode populate_log* pada sistem *Lup Viewer*. *Pseudocode populate_log* dirancang untuk membaca seluruh rekaman mahasiswa, dari setiap rekaman diambil nilai-nilai yang dibutuhkan di antaranya adalah waktu rekaman, baris yang ditambah, dan baris yang dihapus.

No	populate_logs
1	START 2 CALL log_model RETURNING student_records 3 4 FOR each record in student_records 5 CALL main_ctrl with relativize_datetime RETURNING 6 ↳ relative_time 7 GET time 8 GET sha 9 SET insertion to 0

```

9      SET deletion to 0
10     IF selected_file is true
11       IF CALL log_model with is_exist RETURNING true
12         GET insertion
13         GET deletion
14       ENDIF
15     ENDIF
16   ENDFOR
17   SET log to (relative_time, time, sha, insertion,
18   ↳ deletion)
18   RETURN log
19 END

```

Tabel Kode 5.6 Pseudocode function *populate_logs*

5.1.2.3 Perancangan Komponen *Class SearchController*

Pada Tabel Kode 5.7 dipaparkan pseudocode *construct_ed_graph_path* pada sistem *Lup Viewer*. Pseudocode *construct_ed_graph_path* dirancang untuk membangun *path* untuk nama grafik *edit-distance* yang disimpan. *Path* tersebut mengikuti jumlah mahasiswa yang ada. Jika terdapat lebih dari satu mahasiswa, maka diberikan “_” sebagai penghubung nama antar mahasiswa.

No	construct_ed_graph_path
1	<pre> START CALL main_model RETURNING record_path 3 4 IF passed argument == 1 5 SET graph_fmt to argument + ".png" 6 ELSE 7 SET graph_fmt to argument + "_" + ".png" 8 ENDIF 9 SET graph_path record_path + "lupv_notes" + graph_fmt 10 RETURN graph_path 11 END </pre>

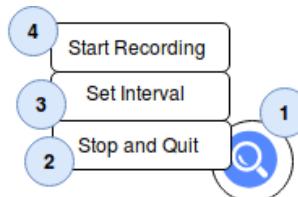
Tabel Kode 5.7 Pseudocode function *construct_ed_graph_path*

5.1.3 Perancangan Antarmuka

Perancangan antarmuka merupakan proses merancang antarmuka sistem dengan menggambar gambaran dasar antarmuka sistem. Rancangan antarmuka dibuat dengan bantuan kakas bantu *draw.io*. Gambaran rancangan ini nantinya dijadikan landasan implementasi antarmuka. Terdapat tiga sampel perancangan antarmuka yang akan dipaparkan, yaitu perancangan antarmuka tampilan *tray* sistem *Lup Recorder*, perancangan antarmuka tampilan hasil rekaman sistem *Lup Viewer*, dan perancangan antarmuka tampilan *edit-distance* tugas sebelumnya.

5.1.3.1 Perancangan Antarmuka Tampilan *Tray* Pada Sistem *Lup Recorder*

Gambar 5.22 merupakan rancangan antarmuka tampilan *tray* pada sistem *Lup Recorder*. Antarmuka ini merupakan rancangan antarmuka utama pada sistem *Lup Recorder* dan digunakan untuk mengakses seluruh menu utama sistem *Lup Recorder*. Penjelasan komponen-komponen yang terdapat pada rancangan tersebut terdapat pada Tabel 5.23.



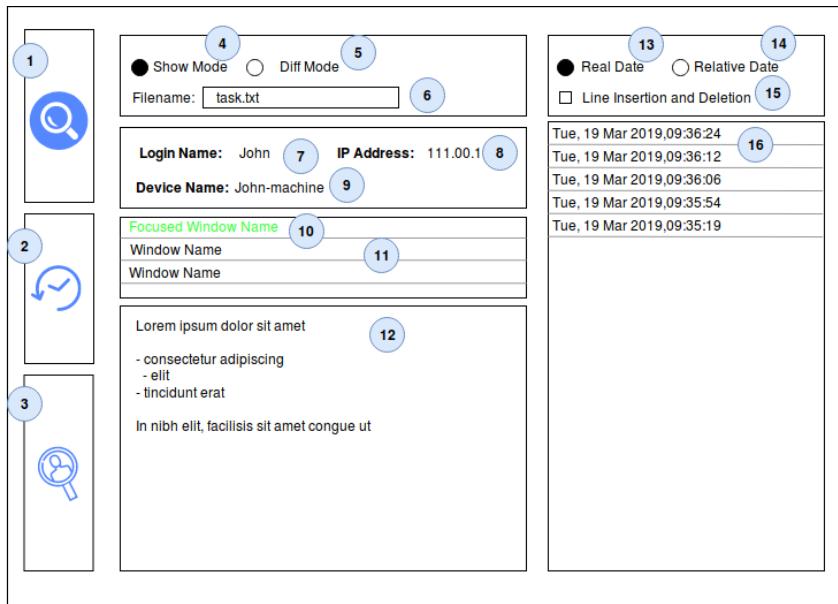
Gambar 5.22 Perancangan antarmuka tampilan *Tray* sistem *Lup Recorder*

Tabel 5.23 Perancangan antarmuka tampilan *Tray* sistem *Lup Recorder*

No	Nama Objek	Tipe	Keterangan
1	<i>Lup Recorder</i> tray	Tombol tray	Tombol untuk mengakses menu pada sistem <i>Lup Recorder</i> .
2	Tombol “Start Recording”	Tombol	Tombol memulai merekam tugas.
3	Tombol “Set Interval”	Tombol	Tombol mengubah interval rekaman.
4	Tombol “Stop and Quit”	Tombol	Tombol untuk menghentikan perekaman tugas.

5.1.3.2 Perancangan Antarmuka Tampilan Hasil Rekaman Pada Sistem *Lup Viewer*

Gambar 5.23 merupakan rancangan antarmuka tampilan hasil rekaman pada sistem *Lup Viewer*. Antarmuka ini merupakan rancangan antarmuka yang digunakan untuk melihat hasil rekaman. Penjelasan komponen-komponen yang terdapat pada rancangan tersebut terdapat pada Tabel 5.24.



Gambar 5.23 Perancangan antarmuka tampilan Hasil Rekaman sistem *Lup Viewer*

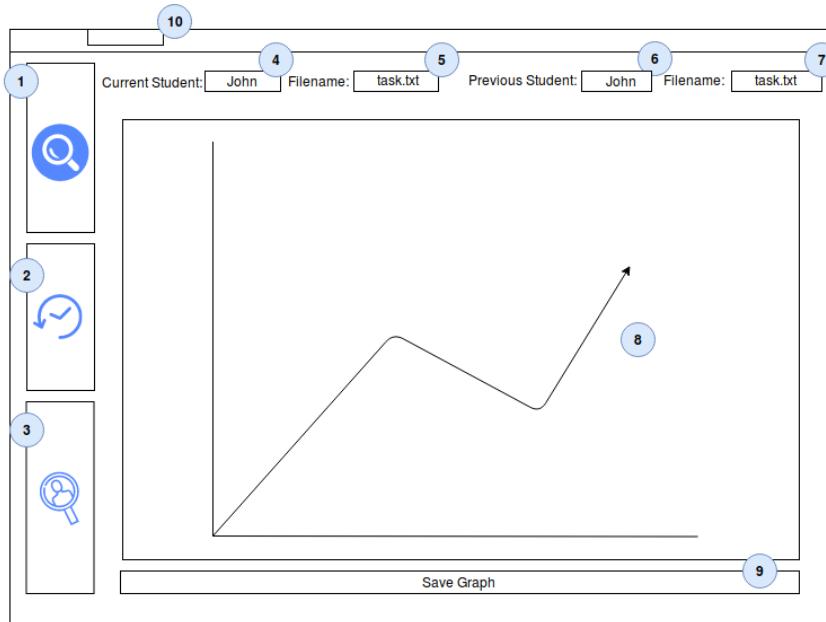
Tabel 5.24 Penjelasan antarmuka tampilan Hasil Rekaman sistem *Lup Viewer*

No	Nama Objek	Tipe	Keterangan
1	Tombol “Main View”	Tombol	Tombol untuk menuju tampilan utama seluruh daftar rekaman.
2	Tombol “Log View”	Tombol	Tombol untuk menuju tampilan hasil rekaman.
3	Tombol “Search View”	Tombol	Tombol untuk menuju tampilan analisis hasil rekaman.
4	Tombol “Show Mode”	Radio Button	<i>Radio button</i> untuk mengalihkan mode tampilan rekaman berkas ke mode <i>show</i> .

5	Tombol “ <i>Diff Mode</i> ”	<i>Radio Button</i>	<i>Radio button</i> untuk mengalihkan mode tampilan rekaman berkas ke mode <i>diff</i> .
6	Tombol “ <i>Filename</i> ”	<i>Combo Box</i>	<i>Combo box</i> untuk memilih nama berkas tugas.
7	Label “ <i>Login Name</i> ”	Label	Label untuk menampilkan nama <i>login</i> .
8	Label “ <i>Device Name</i> ”	Label	Label untuk menampilkan nama piranti.
9	Label “ <i>IP Address</i> ”	Label	Label untuk menampilkan alamat <i>IP</i> .
10	Daftar “ <i>Active Window</i> ”	<i>List Widget</i>	Daftar untuk menampilkan <i>window</i> yang aktif.
11	Daftar “ <i>All Window</i> ”	<i>List Widget</i>	<i>List widget</i> untuk menampilkan semua <i>window</i> .
12	Widget berkas rekaman	Widget	Widget untuk menampilkan rekaman berkas.
13	Tombol “ <i>Real Date</i> ”	<i>Radio Button</i>	<i>Radio button</i> untuk mengalihkan mode format waktu rekaman ke dalam format waktu <i>real</i> .
14	Tombol “ <i>Relative Date</i> ”	<i>Radio Button</i>	<i>Radio button</i> untuk mengalihkan mode format waktu rekaman ke dalam format waktu relatif.
15	Tombol “ <i>Line Insertion and Deletion</i> ”	<i>Check Box</i>	Tombol untuk menampilkan jumlah baris yang ditambah dan dihapus.
16	Daftar Rekaman	<i>List Widget</i>	<i>List widget</i> untuk menampilkan daftar rekaman.

5.1.3.3 Perancangan Antarmuka Tampilan *Edit-distance* Tugas Sebelumnya Pada Sistem *Lup Viewer*

Gambar 5.24 merupakan rancangan antarmuka tampilan *edit-distance* tugas sebelumnya pada sistem *Lup Viewer*. Rancangan antarmuka ini digunakan untuk melihat grafik *edit-distance* tugas sebelumnya. Penjelasan komponen-komponen yang terdapat pada rancangan tersebut terdapat pada Tabel 5.25.



Gambar 5.24 Perancangan antarmuka tampilan *Edit-distance* Tugas Sebelumnya

Tabel 5.25 Penjelasan antarmuka tampilan *Edit-distance* Tugas Sebelumnya

No	Nama Objek	Tipe	Keterangan
1	Tombol “Main View”	Tombol	Tombol untuk menuju tampilan utama seluruh daftar rekaman.
2	Tombol “Log View”	Tombol	Tombol untuk menuju tampilan hasil rekaman.
3	Tombol “Search View”	Tombol	Tombol untuk menuju tampilan analisis hasil rekaman.
4	Tombol “Current Student”	Combo Box	Combo box untuk memilih mahasiswa pada tugas saat ini.
5	Tombol “Filename”	Combo Box	Combo box untuk memilih berkas tugas saat ini.

6	Tombol “ <i>Previous Student</i> ”	<i>Combo Box</i>	<i>Combo box</i> untuk memilih mahasiswa pada tugas sebelumnya.
7	Tombol “ <i>Filename</i> ”	<i>Combo Box</i>	<i>Combo box</i> untuk memilih mahasiswa pada tugas sebelumnya.
8	<i>Widget Edit-distance</i>	<i>Widget</i>	<i>Widget</i> untuk menampilkan grafik <i>edit-distance</i> .
9	Tombol “ <i>Save Graph</i> ”	Tombol	Tombol untuk menyimpan grafik <i>edit-distance</i> .
10	Tombol “ <i>Load Edit-distance File</i> ”	Tombol	Tombol untuk membuka berkas <i>edit-distance</i> tugas sebelumnya.

5.2 Implementasi Sistem

Tahapan implementasi sistem merupakan sebuah tahapan yang dilakukan setelah selesainya proses perancangan sistem. Pada tahapan ini hasil perancangan komponen yang berupa *pseudocode* diimplementasikan menjadi *working code* menggunakan bahasa pemrograman *Python* dan bantuan kakas bantu seperti *git*. Selain itu, Hasil perancangan antarmuka juga diimplementasikan dengan bantuan *widget toolkit Qt*.

5.2.1 Spesifikasi Sistem

Spesifikasi perangkat keras yang digunakan untuk membangun sistem dipaparkan pada Tabel 5.26. Pada Tabel tersebut terdapat spesifikasi *processor*, *HDD* dan *RAM*. Sementara itu, spesifikasi perangkat lunak yang digunakan dipaparkan pada Tabel 5.27, di dalamnya terdapat spesifikasi sistem operasi, *kernel*, editor dokumentasi, editor pemrograman, dan bahasa pemrograman.

Tabel 5.26 Spesifikasi perangkat keras

Nama Komponen	Spesifikasi
<i>Processor</i>	Intel i5-7200U (4) @ 3.1GHz
<i>Hard Disk</i>	500 GB
<i>RAM</i>	4 GB

Tabel 5.27 Spesifikasi perangkat lunak

Nama Komponen	Spesifikasi
Sistem Operasi	Debian GNU/Linux 9.6 (stretch) x86_64
Kernel	Kernel: 4.9.0-6-amd64
Editor Dokumentasi	GNU Emacs 26.1
Editor Pemrograman	GNU Emacs 26.1
Bahasa Pemrograman	Python 3.6.4

5.2.2 Implementasi Kode Program

Pada tahapan implementasi kode program. Hasil rancangan komponen sistem pada tahapan perancangan komponen sistem diimplementasikan menjadi *working code* dengan bahasa pemrograman *Python*. Subbab 5.2.2.1 hingga 5.2.2.3 memaparkan implementasi dari tiga sampel *class* pada tahapan perancangan komponen sebelumnya, yaitu *class Controller* pada sistem *Lup Recorder*, *class LogController* pada sistem *Lup Viewer*, dan *class SearchController* pada sistem *Lup Viewer*.

5.2.2.1 Implementasi Kode Program *Class Controller* Pada Sistem *Lup Recorder*

Pada Tabel Kode 5.8 dipaparkan *source code* *get_all_windows*. *Source code* *get_all_windows* mengimplementasikan hasil rancangan *pseudocode* pada tahapan sebelumnya. *Source code* *get_all_windows* mengambil seluruh informasi *window* yang kemudian melakukan filter dan hanya mengembalikan nama *window* saja.

No	get_all_windows
1	<pre> def get_all_windows(self): all_windows = "" all_windows_proc = Popen(["wmctrl", "-l"], stdout=PIPE) all_windows_dirty, err = all_windows_proc.communicate() if all_windows_dirty: all_windows_dirty = all_windows_dirty.splitlines() for line in all_windows_dirty: windows_name = line.split(None, 3)[-1].decode() </pre>

9	all_windows += " {} \n".format(windows_name)
10	return all_windows

Tabel Kode 5.8 Source code *function get_all_windows*

5.2.2.2 Implementasi Kode Program *Class LogController* Pada Sistem *Lup View*

Pada Tabel Kode 5.9 dipaparkan *source code populate_log*. *Source code populate_log* mengimplementasikan hasil rancangan *pseudocode* pada tahapan sebelumnya. *Source code populate_log* membaca seluruh rekaman mahasiswa, dari setiap rekaman diambil nilai-nilai yang dibutuhkan di antaranya adalah waktu rekaman, baris yang ditambah, dan baris yang dihapus.

No	populate_logs
	<pre> 1 def populate_logs(self, selected_file=None): 2 student_records = self._log_model.student_records 3 4 for record in student_records: 5 relative_time = self._main_ctrl.relativize_datetime(6 record.committed_datetime 7) 8 time_format = "{:%a, %d %b %Y, %H:%M:%S}" 9 time = time_format.format(record.committed_datetime) 10 sha = record.hexsha 11 insertions = 0 12 deletions = 0 13 14 if selected_file: 15 if self._log_model.is_exists(selected_file, sha): 16 insertions = 17 → record.stats.files[selected_file]["insertions"] 18 deletions = 19 → record.stats.files[selected_file]["deletions"] 20 21 log = dict(22 relative_time=relative_time, 23 time=time, 24 sha=sha, 25 insertions=insertions, 26 deletions=deletions, 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 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 159 160 161 162 163 164 165 166 167 168 169 169 170 171 172 173 174 175 176 177 178 179 179 180 181 182 183 184 185 186 187 188 189 189 190 191 192 193 194 195 196 197 198 199 199 200 201 202 203 204 205 206 207 208 209 209 210 211 212 213 214 215 216 217 217 218 219 219 220 221 222 223 224 225 226 227 227 228 229 229 230 231 232 233 234 235 235 236 237 237 238 239 239 240 241 242 243 243 244 245 245 246 247 247 248 248 249 249 250 250 251 251 252 252 253 253 254 254 255 255 256 256 257 257 258 258 259 259 260 260 261 261 262 262 263 263 264 264 265 265 266 266 267 267 268 268 269 269 270 270 271 271 272 272 273 273 274 274 275 275 276 276 277 277 278 278 279 279 280 280 281 281 282 282 283 283 284 284 285 285 286 286 287 287 288 288 289 289 290 290 291 291 292 292 293 293 294 294 295 295 296 296 297 297 298 298 299 299 300 300 301 301 302 302 303 303 304 304 305 305 306 306 307 307 308 308 309 309 310 310 311 311 312 312 313 313 314 314 315 315 316 316 317 317 318 318 319 319 320 320 321 321 322 322 323 323 324 324 325 325 326 326 327 327 328 328 329 329 330 330 331 331 332 332 333 333 334 334 335 335 336 336 337 337 338 338 339 339 340 340 341 341 342 342 343 343 344 344 345 345 346 346 347 347 348 348 349 349 350 350 351 351 352 352 353 353 354 354 355 355 356 356 357 357 358 358 359 359 360 360 361 361 362 362 363 363 364 364 365 365 366 366 367 367 368 368 369 369 370 370 371 371 372 372 373 373 374 374 375 375 376 376 377 377 378 378 379 379 380 380 381 381 382 382 383 383 384 384 385 385 386 386 387 387 388 388 389 389 390 390 391 391 392 392 393 393 394 394 395 395 396 396 397 397 398 398 399 399 400 400 401 401 402 402 403 403 404 404 405 405 406 406 407 407 408 408 409 409 410 410 411 411 412 412 413 413 414 414 415 415 416 416 417 417 418 418 419 419 420 420 421 421 422 422 423 423 424 424 425 425 426 426 427 427 428 428 429 429 430 430 431 431 432 432 433 433 434 434 435 435 436 436 437 437 438 438 439 439 440 440 441 441 442 442 443 443 444 444 445 445 446 446 447 447 448 448 449 449 450 450 451 451 452 452 453 453 454 454 455 455 456 456 457 457 458 458 459 459 460 460 461 461 462 462 463 463 464 464 465 465 466 466 467 467 468 468 469 469 470 470 471 471 472 472 473 473 474 474 475 475 476 476 477 477 478 478 479 479 480 480 481 481 482 482 483 483 484 484 485 485 486 486 487 487 488 488 489 489 490 490 491 491 492 492 493 493 494 494 495 495 496 496 497 497 498 498 499 499 500 500 501 501 502 502 503 503 504 504 505 505 506 506 507 507 508 508 509 509 510 510 511 511 512 512 513 513 514 514 515 515 516 516 517 517 518 518 519 519 520 520 521 521 522 522 523 523 524 524 525 525 526 526 527 527 528 528 529 529 530 530 531 531 532 532 533 533 534 534 535 535 536 536 537 537 538 538 539 539 540 540 541 541 542 542 543 543 544 544 545 545 546 546 547 547 548 548 549 549 550 550 551 551 552 552 553 553 554 554 555 555 556 556 557 557 558 558 559 559 560 560 561 561 562 562 563 563 564 564 565 565 566 566 567 567 568 568 569 569 570 570 571 571 572 572 573 573 574 574 575 575 576 576 577 577 578 578 579 579 580 580 581 581 582 582 583 583 584 584 585 585 586 586 587 587 588 588 589 589 590 590 591 591 592 592 593 593 594 594 595 595 596 596 597 597 598 598 599 599 600 600 601 601 602 602 603 603 604 604 605 605 606 606 607 607 608 608 609 609 610 610 611 611 612 612 613 613 614 614 615 615 616 616 617 617 618 618 619 619 620 620 621 621 622 622 623 623 624 624 625 625 626 626 627 627 628 628 629 629 630 630 631 631 632 632 633 633 634 634 635 635 636 636 637 637 638 638 639 639 640 640 641 641 642 642 643 643 644 644 645 645 646 646 647 647 648 648 649 649 650 650 651 651 652 652 653 653 654 654 655 655 656 656 657 657 658 658 659 659 660 660 661 661 662 662 663 663 664 664 665 665 666 666 667 667 668 668 669 669 670 670 671 671 672 672 673 673 674 674 675 675 676 676 677 677 678 678 679 679 680 680 681 681 682 682 683 683 684 684 685 685 686 686 687 687 688 688 689 689 690 690 691 691 692 692 693 693 694 694 695 695 696 696 697 697 698 698 699 699 700 700 701 701 702 702 703 703 704 704 705 705 706 706 707 707 708 708 709 709 710 710 711 711 712 712 713 713 714 714 715 715 716 716 717 717 718 718 719 719 720 720 721 721 722 722 723 723 724 724 725 725 726 726 727 727 728 728 729 729 730 730 731 731 732 732 733 733 734 734 735 735 736 736 737 737 738 738 739 739 740 740 741 741 742 742 743 743 744 744 745 745 746 746 747 747 748 748 749 749 750 750 751 751 752 752 753 753 754 754 755 755 756 756 757 757 758 758 759 759 760 760 761 761 762 762 763 763 764 764 765 765 766 766 767 767 768 768 769 769 770 770 771 771 772 772 773 773 774 774 775 775 776 776 777 777 778 778 779 779 780 780 781 781 782 782 783 783 784 784 785 785 786 786 787 787 788 788 789 789 790 790 791 791 792 792 793 793 794 794 795 795 796 796 797 797 798 798 799 799 800 800 801 801 802 802 803 803 804 804 805 805 806 806 807 807 808 808 809 809 810 810 811 811 812 812 813 813 814 814 815 815 816 816 817 817 818 818 819 819 820 820 821 821 822 822 823 823 824 824 825 825 826 826 827 827 828 828 829 829 830 830 831 831 832 832 833 833 834 834 835 835 836 836 837 837 838 838 839 839 840 840 841 841 842 842 843 843 844 844 845 845 846 846 847 847 848 848 849 849 850 850 851 851 852 852 853 853 854 854 855 855 856 856 857 857 858 858 859 859 860 860 861 861 862 862 863 863 864 864 865 865 866 866 867 867 868 868 869 869 870 870 871 871 872 872 873 873 874 874 875 875 876 876 877 877 878 878 879 879 880 880 881 881 882 882 883 883 884 884 885 885 886 886 887 887 888 888 889 889 890 890 891 891 892 892 893 893 894 894 895 895 896 896 897 897 898 898 899 899 900 900 901 901 902 902 903 903 904 904 905 905 906 906 907 907 908 908 909 909 910 910 911 911 912 912 913 913 914 914 915 915 916 916 917 917 918 918 919 919 920 920 921 921 922 922 923 923 924 924 925 925 926 926 927 927 928 928 929 929 930 930 931 931 932 932 933 933 934 934 935 935 936 936 937 937 938 938 939 939 940 940 941 941 942 942 943 943 944 944 945 945 946 946 947 947 948 948 949 949 950 950 951 951 952 952 953 953 954 954 955 955 956 956 957 957 958 958 959 959 960 960 961 961 962 962 963 963 964 964 965 965 966 966 967 967 968 968 969 969 970 970 971 971 972 972 973 973 974 974 975 975 976 976 977 977 978 978 979 979 980 980 981 981 982 982 983 983 984 984 985 985 986 986 987 987 988 988 989 989 990 990 991 991 992 992 993 993 994 994 995 995 996 996 997 997 998 998 999 999 1000 1000 1001 1001 1002 1002 1003 1003 1004 1004 1005 1005 1006 1006 1007 1007 1008 1008 1009 1009 1010 1010 1011 1011 1012 1012 1013 1013 1014 1014 1015 1015 1016 1016 1017 1017 1018 1018 1019 1019 1020 1020 1021 1021 1022 1022 1023 1023 1024 1024 1025 1025 1026 1026 1027 1027 1028 1028 1029 1029 1030 1030 1031 1031 1032 1032 1033 1033 1034 1034 1035 1035 1036 1036 1037 1037 1038 1038 1039 1039 1040 1040 1041 1041 1042 1042 1043 1043 1044 1044 1045 1045 1046 1046 1047 1047 1048 1048 1049 1049 1050 1050 1051 1051 1052 1052 1053 1053 1054 1054 1055 1055 1056 1056 1057 1057 1058 1058 1059 1059 1060 1060 1061 1061 1062 1062 1063 1063 1064 1064 1065 1065 1066 1066 1067 1067 1068 1068 1069 1069 1070 1070 1071 1071 1072 1072 1073 1073 1074 1074 1075 1075 1076 1076 1077 1077 1078 1078 1079 1079 1080 1080 1081 1081 1082 1082 1083 1083 1084 1084 1085 1085 1086 1086 1087 1087 1088 1088 1089 1089 1090 1090 1091 1091 1092 1092 1093 1093 1094 1094 1095 1095 1096 1096 1097 1097 1098 1098 1099 1099 1100 1100 1101 1101 1102 1102 1103 1103 1104 1104 1105 1105 1106 1106 1107 1107 1108 1108 1109 1109 1110 1110 1111 1111 1112 1112 1113 1113 1114 1114 1115 1115 1116 1116 1117 1117 1118 1118 1119 1119 1120 1120 1121 1121 1122 1122 1123 1123 1124 1124 1125 1125 1126 1126 1127 1127 1128 1128 1129 1129 1130 1130 1131 1131 1132 1132 1133 1133 1134 1134 1135 1135 1136 1136 1137 1137 1138 1138 1139 1139 1140 1140 1141 1141 1142 1142 1143 1143 1144 1144 1145 1145 1146 1146 1147 1147 1148 1148 1149 1149 1150 1150 1151 1151 1152 1152 1153 1153 1154 1154 1155 1155 1156 1156 1157 1157 1158 1158 1159 1159 1160 1160 1161 1161 1162 1162 1163 1163 1164 1164 1165 1165 1166 1166 1167 1167 1168 1168 1169 1169 1170 1170 1171 1171 1172 1172 1173 1173 1174 1174 1175 1175 1176 1176 1177 1177 1178 1178 1179 1179 1180 1180 1181 1181 1182 1182 1183 1183 1184 1184 1185 1185 1186 1186 1187 1187 1188 1188 1189 1189 1190 1190 1191 1191 1192 1192 1193 1193 1194 1194 1195 1195 1196 1196 1197 1197 1198 1198 1199 1199 1200 1200 1201 1201 1202 1202 1203 1203 1204 1204 1205 1205 1206 1206 1207 1207 1208 1208 1209 1209 1210 1210 1211 1211 1212 1212 1213 1213 1214 1214 1215 1215 1216 1216 1217 1217 1218 1218 1219 1219 1220 1220 1221 1221 1222 1222 1223 1223 1224 1224 1225 1225 1226 1226 1227 1227 1228 1228 1229 1229 1230 1230 1231 1231 1232 1232 1233 1233 1234 1234 1235 1235 1236 1236 1237 1237 1238 1238 1239 1239 1240 1240 1241 1241 1242 1242 1243 1243 1244 1244 1245 1245 1246 1246 1247 1247 1248 1248 1249 1249 1250 1250 1251 1251 1252 1252 1253 1253 1254 1254 1255 1255 1256 1256 1257 1257 1258 1258 1259 1259 1260 1260 1261 1261 1262 1262 1263 1263 1264 1264 1265 1265 1266 1266 1267 1267 1268 1268 1269 1269 1270 1270 1271 1271 1272 1272 1273 1273 1274 1274 1275 1275 1276 1276 1277 1277 1278 1278 1279 1279 1280 1280 1281 1281 1282 1282 1283 1283 1284 1284 1285 1285 1286 1286 1287 1287 1288 1288 1289 1289 1290 1290 1291 1291 1292 1292 1293 1293 1294 1294 1295 1295 1296 1296 1297 1297 1298 1298 1299 1299 1300 1300 1301 1301 1302 1302 1303 1303 1304 1304 1305 1305 1306 1306 1307 1307 1308 1308 1309 1309 1310 1310 1311 1311 1312 1312 1313 1313 1314 1314 1315 1315 1316 1316 1317 1317 1318 1318 1319 1319 1320 1320 1321 1321 1322 1322 1323 1323 1324 1324 1325 1325 1326 1326 1327 1327 1328 1328 1329 1329 1330 1330 1331 1331 1332 1332 1333 1333 1334 1334 1335 1335 1336 1336 1337 1337 1338 1338 1339 1339 1340 1340 1341 1341 1342 1</pre>

25)
26	yield log

Tabel Kode 5.9 Source code *function populate_logs*

5.2.2.3 Implementasi Kode Program *Class SearchController* Pada Sistem *Lup Viewer*

Pada Tabel Kode 5.9 dipaparkan *source code construct_ed_graph_path*. *Source code construct_ed_graph_path* mengimplementasikan hasil rancangan *pseudocode* pada tahapan sebelumnya. *Source code construct_ed_graph_path* membangun *path* untuk nama grafik *edit-distance* yang disimpan. *Path* tersebut mengikuti jumlah mahasiswa yang ada. Jika terdapat lebih dari satu mahasiswa, maka diberikan “_” sebagai penghubung nama antar mahasiswa.

No	construct_ed_graph_path
1	def construct_ed_graph_path (self, *args): record_path = self._main_model.record_path if len(args) == 1: graph_fmt = "{}.png".format(args[0]) else : names = "_".join(args) graph_fmt = "{}.png".format(names) graph_path = join(record_path, "lupv-notes", graph_fmt) return graph_path

Tabel Kode 5.10 Source code *function construct_ed_graph_path*

5.2.3 Implementasi Antarmuka

Pada Tahapan ini, hasil rancangan antarmuka pada tahapan sebelumnya diimplementasikan menggunakan *widget toolkit Qt*. Tiga sampel implementasi antarmuka yang dipaparkan adalah antarmuka tampilan *tray* pada sistem *Lup Recorder* pada Gambar 5.25, antarmuka tampilan hasil rekaman pada sistem *Lup Viewer* pada Gambar 5.26, dan antarmuka tampilan *edit-distance* tugas sebelumnya pada sistem *Lup Viewer* pada Gambar 5.27.

5.2.3.1 Implementasi Antarmuka Tampilan *Tray* Pada Sistem *Lup Recorder*

Gambar 5.25 merupakan implementasi antarmuka tampilan *tray* pada sistem *Lup Recorder*. Implementasi ini dibangun berlandaskan tahapan perancangan antarmuka sebelumnya. Antarmuka ini merupakan rancangan antarmuka utama pada sistem *Lup Recorder* dan digunakan untuk mengakses seluruh menu utama sistem *Lup Recorder*.



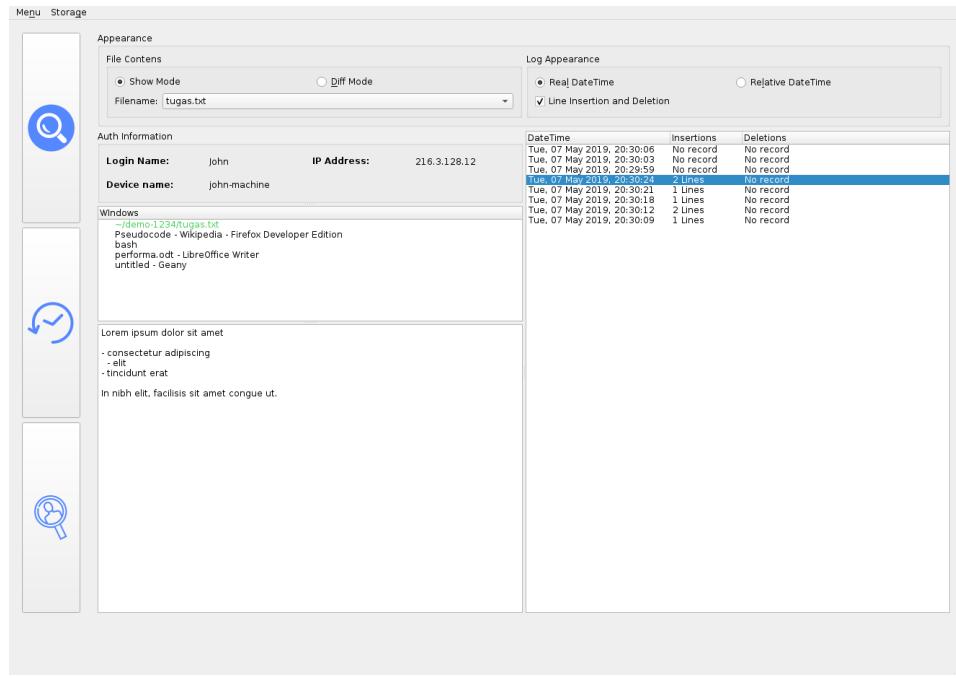
Gambar 5.25 Implementasi antarmuka tampilan *Tray* pada sistem *Lup Recorder*

5.2.3.2 Implementasi Antarmuka Tampilan Hasil Rekaman Pada Sistem *Lup Viewer*

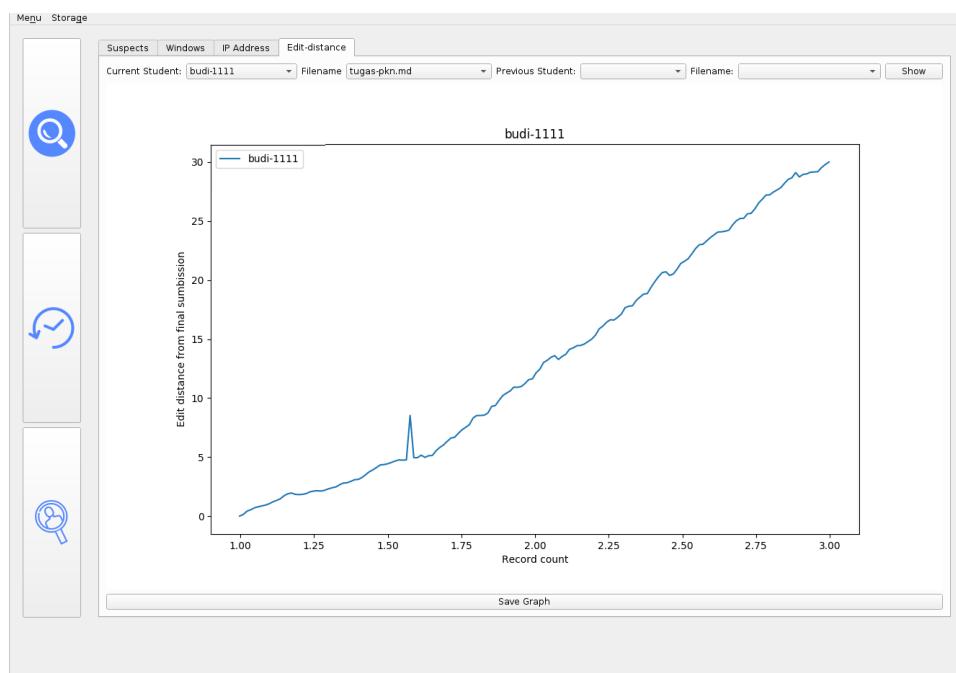
Gambar 5.26 merupakan implementasi antarmuka tampilan hasil rekaman pada sistem *Lup Viewer*. Implementasi ini dibangun berlandaskan tahapan perancangan antarmuka sebelumnya. Antarmuka ini merupakan rancangan antarmuka yang digunakan untuk melihat hasil rekaman.

5.2.3.3 Perancangan Antarmuka Tampilan *Edit-distance* Tugas Sebelumnya Pada Sistem *Lup Viewer*

Gambar 5.27 merupakan implementasi antarmuka tampilan *edit-distance* tugas sebelumnya pada sistem *Lup Viewer*. Implementasi ini dibangun berlandaskan tahapan perancangan antarmuka sebelumnya. Antarmuka ini digunakan untuk melihat grafik *edit-distance* tugas sebelumnya.



Gambar 5.26 Implementasi antarmuka tampilan Hasil Rekaman pada sistem *Lup Viewer*



Gambar 5.27 Perancangan antarmuka tampilan *Edit-distance* Tugas Sebelumnya pada sistem *Lup Viewer*

BAB 6 PENGUJIAN

6.1 Pengujian Unit

Pengujian unit dilakukan untuk memastikan hasil implementasi kode program telah sesuai dengan hasil perancangan komponen. Komponen terkecil dari sebuah sistem berorientasi objek adalah *class* dengan *testable unit* terkecilnya adalah *function* atau *method* (Pressman, 2010). Pengujian unit pada *class model* dan *class controller* dilakukan hingga mencapai 100% *code coverage*. *Class view* hanya memanggil fungsi-fungsi yang terdapat pada *class controller* dan *class model* sehingga pada penelitian ini pengujian unit pada *class view* hanya diuji hingga *code coverage* mencapai 70%. Metode pengujian yang digunakan adalah *white-box testing*, dan teknik pengujian yang digunakan adalah *basis path testing*. *Driver* digunakan sebagai “*main program*” untuk memanggil *function* yang diuji. *Stub* atau *fake* digunakan agar pengujian unit benar-benar terisolasi, bukan sebagai “*stand-alone program*”. Terdapat tiga sampel pengujian unit yang akan dipaparkan pada tahapan ini. Gambar 6.28 memperlihatkan hasil pengujian unit pada *class controller* dan *class model* yang mencapai 100% *code coverage*.

Name	Stmts	Miss	Cover
<hr/>			
Lupv/__init__.py	0	0	100%
Lupv/controllers/__init__.py	0	0	100%
Lupv/controllers/log.py	64	0	100%
Lupv/controllers/main.py	54	0	100%
Lupv/controllers/search.py	172	0	100%
Lupv/models/__init__.py	0	0	100%
Lupv/models/logs.py	75	0	100%
Lupv/models/main.py	42	0	100%
Lupv/models/search.py	30	0	100%
<hr/>			
TOTAL	437	0	100%
===== 91 passed in 6.45 seconds			

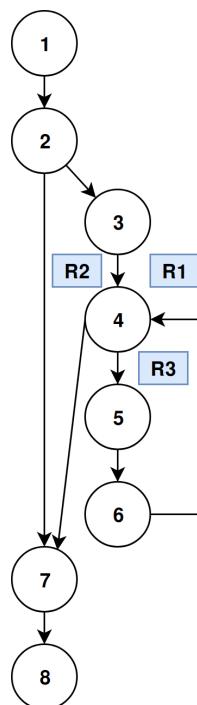
Gambar 6.28 Pengujian unit mencapai 100% *code coverage*

6.1.1 Pengujian Unit *Class Controller Function get_all_windows*

Pengujian *unit* dilakukan pada *function get_all_windows* secara terisolasi. Maka unit lain yang dipanggil di dalamnya akan digantikan dengan *stub*. Tabel Kode 6.11 dan Gambar 6.29 menjelaskan *basis path* dari *pseudocode function get_all_windows*. Bagian berikutnya menjelaskan perhitungan *cyclomatic complexity*, dan *independent path*. Tabel 6.28 menjelaskan hasil uji dan kasus uji dari *function get_all_windows*.

No	get_all_windows
1	START
2	READ all_windows_dirty
3	IF all_windows_dirty is true
4	SET all_windows_dirty to split every line
5	FOR each window in all_windows_dirty
6	GET window_name
7	SET all_windows = window_name
8	ENDFOR
9	ENDIF
10	RETURN all_windows
11	END

Tabel Kode 6.11 *Pseudocode function get_all_windows*



Gambar 6.29 *Flow graph* dari *pseudocode get_all_windows*

Cyclomatic Complexity

- $V(G) = 3$ regions
- $V(G) = 9 \text{ edge} - 8 \text{ node} + 2 = 3$
- $V(G) = 2 \text{ predicate node} + 1 = 3$

Independent Path

- Jalur 1 : 1 - 2 - 7 - 8
- Jalur 2 : 1 - 2 - **3 - 4** - 7 - 8
- Jalur 2 : 1 - 2 - 3 - 4 - **5 - 6** - 4 - 7 - 8

Tabel 6.28 Kasus uji dan hasil uji *function get_all_windows*

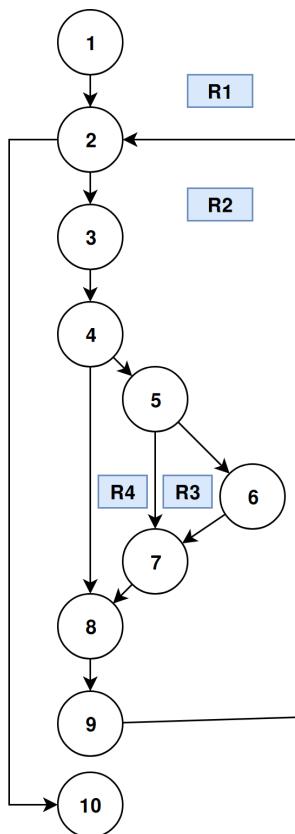
Jalur	Prosedur Uji	Expected Result	Result	Status
1	1) Set variable <i>window_title</i> dengan nilai "None". 2) Class Driver <i>TestController</i> memanggil function <i>get_all_windows</i>	<i>Return value function get_all_windows bernilai "'''".</i>	As expected	Valid
2	1) Set variable <i>window_title</i> dengan nilai "'''". 2) Class Driver <i>TestController</i> memanggil function <i>get_all_windows</i> .	<i>Return value function get_all_windows bernilai "'''".</i>	As expected	Valid
3	1) Set variable <i>window_title</i> dengan nilai "b"0x006000ab 0 machine-name <i>foo_window_title</i> "". 2) Class Driver <i>TestController</i> memanggil function <i>get_all_windows</i> .	<i>Return value function get_all_windows bernilai "foo_window_title"</i>	As expected	Valid

Pengujian unit yang dilakukan pada *function get_all_windows* menghasilkan perhitungan *cyclomatic complexity* dengan nilai 3 sehingga memiliki 3 *independent path* yang harus diuji. Hasil pengujian 3 *path* yang dipaparkan pada Tabel 6.28 bernilai valid.

6.1.2 Pengujian Unit Class LogController Function `populate_logs`

Pengujian unit dilakukan pada *function populate_logs* secara terisolasi. Maka *function student_record*, *function is_exist* pada unit *LogModel*, dan *function relativize_datetime* pada unit *MainController* yang dipanggil di dalamnya akan digantikan dengan *stub*. Tabel Kode 6.12 dan Gambar 6.30 menjelaskan *basis path* dari *pseudocode function populate_logs*. Bagian berikutnya menjelaskan perhitungan *cyclomatic complexity* dan *independent path*. Tabel 6.29 menjelaskan hasil uji dan kasus uji dari *function populate_logs*.

Tabel Kode 6.12 Pseudocode function populate_logs



Gambar 6.30 Flow graph dari pseudocode *populate_logs*

Cyclomatic Complexity

- $V(G) = 4 \text{ regions}$
- $V(G) = 12 \text{ edge} - 10 \text{ node} + 2 = 4$
- $V(G) = 3 \text{ predicate node} + 1 = 4$

Independent Path

- Jalur 1: 1 - 2 - 10
- Jalur 2: 1 - 2 - 3 - 4 - 8 - 9 - 2 - 10
- Jalur 3: 1 - 2 - 3 - 4 - 5 - 7 - 8 - 9 - 2 - 10
- Jalur 4: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 2 - 10

Tabel 6.29 Kasus uji dan hasil uji *function populate_logs*

Jalur	Prosedur Uji	Expected Result	Result	Status
1	1) Set variable <i>student_records</i> dengan nilai “[]”. 2) Class Driver <i>TestLogController</i> memanggil function <i>populate_logs</i> .	<i>Return value function populate_logs</i> bernilai “[]”.	As expected	Valid
2	Class Driver <i>TestLogController</i> memanggil function <i>populate_logs</i> tanpa argument.	<i>Return value function populate_logs</i> bernilai data rekaman tanpa nilai baris yang ditambah dan dihapus.	As expected	Valid
3	Class Driver <i>TestLogController</i> memanggil function <i>populate_logs</i> dengan argument “selected_file=“tugas-none.txt””.	<i>Return value function populate_logs</i> bernilai data rekaman tanpa nilai baris yang ditambah dan dihapus.	As expected	Valid
4	Class Driver <i>TestLogController</i> memanggil function <i>populate_logs</i> dengan argument “selected_file=“tugas-tif.txt””.	<i>Return value function populate_logs</i> bernilai data rekaman dengan nilai baris yang ditambah dan dihapus.	As expected	Valid

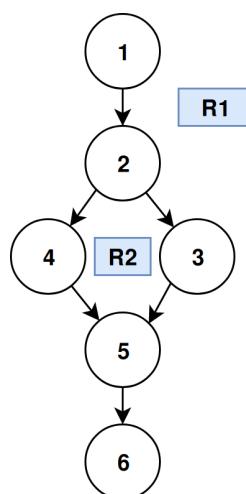
Pengujian unit yang dilakukan pada *function populate_logs* menghasilkan perhitungan *cyclomatic complexity* dengan nilai 4 sehingga memiliki 4 *independent path* yang harus diuji. Hasil pengujian 4 *path* yang dipaparkan pada Tabel 6.29 ber nilai valid.

6.1.3 Pengujian Unit *Class SearchController Function construct_ed_graph_path*

Pengujian *unit* dilakukan pada *function construct_ed_graph_path* secara terisolasi. Maka unit *main_model* yang dipanggil di dalamnya akan digantikan dengan *stub*. Tabel Kode 6.13 dan Gambar 6.31 menjelaskan *basis path* dari *pseudocode function construct_ed_graph_path*. Bagian berikutnya menjelaskan perhitungan *cyclomatic complexity*, dan *independent path*. Tabel 6.30 menjelaskan hasil uji dan kasus uji dari *function construct_ed_graph_path*.

No	construct_ed_graph_path
1	START
2	CALL main_model RETURNING record_path
3	
4	IF passed argument == 1
5	SET graph_fmt to argument + ".png"
6	ELSE
7	SET graph_fmt to argument + "_" + ".png"
8	ENDIF
9	SET graph_path record_path + "lupv_notes" \
10	+ graph_fmt
11	Return graph_path
12	END

Tabel Kode 6.13 *Pseudocode function construct_ed_graph_path*



Gambar 6.31 *Flow graph* dari *pseudocode construct_ed_graph_path*

Cyclomatic Complexity

- $V(G) = 2 \text{ regions}$
- $V(G) = 6 \text{ edge} - 6 \text{ node} + 2 = 2$
- $V(G) = 1 \text{ predicate node} + 1 = 2$

Independent Path

- Jalur 1: 1 - 2 - 3 - 5 - 6
- Jalur 2: 1 - 2 - 4 - 5 - 6

Tabel 6.30 Kasus uji dan hasil uji *function construct_ed_graph_path*

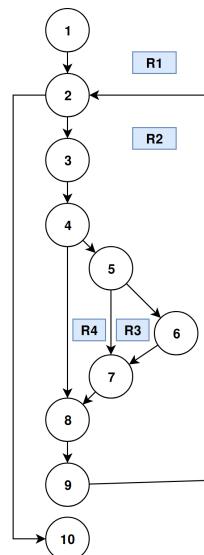
Jalur	Prosedur Uji	Expected Result	Result	Status
1	<i>Class Driver TestSearchController</i> memanggil <i>function construct_ed_graph_path</i> dengan <i>argument</i> "ani-1111".	<i>Return value function construct_ed_graph_path</i> bernilai "/lupv-notes/ani-1111.png".	As expected	Valid
2	<i>Class Driver TestSearchController</i> memanggil <i>function construct_ed_graph_path</i> dengan <i>argument</i> "ani-1111", "budi-2222".	<i>Return value function construct_ed_graph_path</i> bernilai "/lupv-notes/ani-1111_budi-2222.png".	As expected	Valid

Pengujian unit yang dilakukan pada *function construct_ed_graph_path* menghasilkan perhitungan *cyclomatic complexity* dengan nilai 2 sehingga memiliki 2 *independent path* yang harus diuji. Hasil pengujian 2 *path* yang dipaparkan pada Tabel 6.30 bernilai valid.

6.2 Pengujian Integrasi

Pengujian integrasi dilakukan untuk memastikan integrasi antar unit berjalan dengan baik. Pengujian integrasi dilakukan karena terdapat kemungkinan terjadi masalah ketika unit dijalankan bersamaan. Sistem yang dikembangkan menggunakan pendekatan dan pengembangan berorientasi objek sehingga pendekatan pengujian integrasi yang digunakan adalah *use-based testing*. Pengujian integrasi diawali dari *class* yang memiliki *dependent class* paling sedikit, yaitu dimulai dari *class model*, kemudian *class controller*, dan yang terakhir adalah *class view*. Metode yang digunakan adalah *white-box testing*, dan teknik yang digunakan adalah *basis path testing*. Tujuan pengujian integrasi memastikan integrasi komponen sistem berjalan dengan baik. Maka pada pengujian ini, *function* pada suatu *class* yang memanggil *function* pada *class* lain tidak digantikan dengan *stub* atau *fake* sebagaimana ketika pada pengujian unit. Terdapat satu sampel pengujian integrasi yang dipaparkan, yaitu *function populate_logs* pada *class LogController* yang di dalamnya memanggil *function student_records* dan *function is_exist* pada *class LogModel*, dan *function relativize_datetime* pada *class MainController*.

Tabel Kode 6.14 dan Gambar 6.32 menjelaskan *basis path* dari *pseudocode function populate_logs_integration*. Bagian berikutnya menjelaskan perhitungan *cyclomatic complexity*, *independent path*. Tabel 6.31 menjelaskan hasil uji dan kasus uji dari *function populate_logs*.



Gambar 6.32 Flow graph dari pseudocode *populate_logs*

No	populate_logs
1	START
2	CALL log_model RETURNING student_records ————— 1
3	
4	FOR each record in student_records ————— 2
5	CALL main_ctrl with relativize_datetime \
6	RETURNING relative_time
7	GET time
8	GET sha
9	SET insertion to 0
10	SET deletion to 0
11	IF selected_file is true ————— 4
12	IF CALL log_model with is_exist \
13	RETURNING true ————— 5
14	GET insertion
15	GET deletion
16	ENDIF ————— 6
17	ENDIF ————— 7
18	ENDFOR ————— 8
19	SET log to (relative_time, time, sha,
20	insertion, deletion)
21	RETURN log ————— 9
22	END

Tabel Kode 6.14 Pseudocode function populate_logs

Cyclomatic Complexity

- $V(G) = 4$ regions
- $V(G) = 12$ edge – 10 node + 2 = 4
- $V(G) = 3$ predicate node + 1 = 4

Independent Path

- Jalur 1: 1 - 2 - 10
- Jalur 2: 1 - 2 - 3 - 4 - 8 - 9 - 2 - 10
- Jalur 3: 1 - 2 - 3 - 4 - 5 - 7 - 8 - 9 - 2 - 10
- Jalur 4: 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 2 - 10

Tabel 6.31 Kasus uji dan hasil uji *function populate_logs*

Jalur	Prosedur Uji	Expected Result	Result	Status
1	1) Set variable <i>student_records</i> dengan nilai “[]”. 2) Class Driver <i>TestLogController</i> memanggil function <i>populate_logs</i> .	<i>Return value function populate_logs</i> bernilai “[]”.	As expected	Valid
2	Class Driver <i>TestLogController</i> memanggil function <i>populate_logs</i> tanpa argument.	<i>Return value function populate_logs</i> bernilai data rekaman tanpa nilai baris yang ditambah dan dihapus.	As expected	Valid
3	Class Driver <i>TestLogController</i> memanggil function <i>populate_logs</i> dengan argument “selected_file=“tugas-none.txt””.	<i>Return value function populate_logs</i> bernilai data rekaman tanpa nilai baris yang ditambah dan dihapus.	As expected	Valid
4	Class Driver <i>TestLogController</i> memanggil function <i>populate_logs</i> dengan argument “selected_file=“tugas-tif.txt””.	<i>Return value function populate_logs</i> bernilai data rekaman dengan nilai baris yang ditambah dan dihapus.	As expected	Valid

Pengujian integrasi yang dilakukan pada *function populate_logs* menghasilkan perhitungan *cyclomatic complexity* dengan nilai 4 sehingga memiliki 4 *independent path* yang harus diuji. Hasil pengujian 4 *path* yang dipaparkan pada Tabel 6.31 bernilai valid.

6.3 Pengujian Validasi

Pengujian validasi dilakukan untuk memastikan bahwa sistem yang dibangun sesuai dengan seluruh skenario kebutuhan yang didefinisikan. Pengujian validasi fokus terhadap *user-visible action* dan *user-recognizable output* dari sistem (Pressman, 2010). Dalam pengujian validasi metode pengujian yang digunakan adalah *black-box testing*. Teknik yang digunakan untuk memilih nilai masukan pada pengujian adalah teknik *boundary value analysis* dan *equivalence partitioning*, kedua teknik tersebut digunakan untuk menghindari *rigorous testing* atau mencoba semua nilai *input*.

6.3.1 Pengujian Validasi Merekam Pengeraaan Tugas

Pengujian validasi “Merekam Pengeraaan Tugas” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPR-F-01. Terdapat tiga kasus uji pada pengujian ini. Kasus uji pertama pada Tabel 6.32 memastikan bahwa sistem memulai perekaman tugas, kasus uji kedua pada Tabel 6.34 memastikan bahwa sistem berhenti dan keluar, dan kasus uji ketiga pada Tabel 6.33 memastikan sistem menutup dialog pemilihan direktori tugas

Tabel 6.32 Kasus uji dan hasil uji Merekam Pengeraaan Tugas

Kode Kebutuhan	LUPR-F-01
Nama Kasus Uji	Merekam Pengeraaan Tugas
Prosedur	1. Mengklik tombol “Start Recording”. 2. Memilih direktori tempat tugas. 3. Mengklik tombol “Choose”.
Expected Result	Sistem memulai perekaman pengeraaan tugas dan menampilkan pesan notifikasi “Lup is recording” menandakan bahwa perekaman telah dimulai.
Result	As expected
Status	Valid

Tabel 6.33 Kasus uji dan hasil uji Merekam Pengeraaan Tugas alternatif 1

Kode Kebutuhan	LUPR-F-01
Nama Kasus Uji	Merekam Pengeraaan Tugas alternatif 1
Prosedur	1. Mengklik tombol “Stop and Quit”.
Expected Result	Sistem berhenti dan keluar.

Result	<i>As expected</i>
Status	Valid

Tabel 6.34 Kasus uji dan hasil uji Merekam Pengerajan Tugas alternatif 2

Kode Kebutuhan	LUPR-F-01
Nama Kasus Uji	Merekam Pengerajan Tugas alternatif 2
Prosedur	1. Mengklik tombol “Start Recording”. 2. Memilih direktori tempat tugas. 3. Mengklik tombol “Cancel”.
Expected Result	Sistem menutup dialog pemilihan direktori tugas.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Merekam Pengerajan Tugas” yang memiliki 3 kasus uji yaitu pada Tabel 6.32, Tabel 6.33, dan Tabel 6.34 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Merekam Pengerajan Tugas” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.2 Pengujian Validasi Mengubah Interval Rekaman

Pengujian validasi “Mengubah Interval Rekaman” membutuhkan *range* nilai yang harus dimasukkan penguji untuk melakukan validasi. Agar terhindar dari *rigorous testing* digunakan teknik *boundary value analysis* dan *equivalence partitioning* untuk menentukan nilai yang dimasukkan dalam pengujian. Nilai yang valid adalah semua bilangan bulat di atas 0.

Kasus uji yang didapatkan dengan teknik *equivalence partitioning*:

1. Kasus uji *valid input*: “2”
2. Kasus uji *invalid input*: “kolom kosong”

Kasus uji yang didapatkan dengan teknik *boundary value analysis*:

1. Kasus uji *valid input*: “1”
2. Kasus uji *invalid input*: “0”

Terdapat dua kasus uji dengan teknik *equivalence partitioning*, pertama pada Tabel 6.35 dengan nilai *input* “2” memastikan bahwa sistem berhasil mengubah nilai interval, kedua pada Tabel 6.36 dengan mengosongkan kolom memastikan bahwa sistem menampilkan pesan peringatan. Teknik *boundary value analysis* juga menghasilkan dua kasus uji. Pertama pada Tabel 6.37 dengan nilai *input* “1” memastikan bahwa sistem berhasil mengubah nilai interval, kasus uji kedua pada Tabel 6.38 dengan nilai *input* “0” memastikan bahwa sistem menampilkan pesan peringatan.

Tabel 6.35 Kasus uji dan hasil uji Mengubah Interval Rekaman (*valid input equivalence partitioning*)

Kode Kebutuhan	LUPR-F-02
Nama Kasus Uji	Mengubah Interval Rekaman
Prosedur	1. Mengeklik tombol “Set Interval”. 2. Mengisi nilai 2 pada dialog interval. 3. Mengeklik tombol “Apply”.
Expected Result	Sistem mengubah nilai interval rekaman dan menampilkan pesan notifikasi “Interval changed to 2”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.36 Kasus uji dan hasil uji Mengubah Interval Rekaman alternatif 1 (*invalid input equivalence partitioning*)

Kode Kebutuhan	LUPR-F-02
Nama Kasus Uji	Mengubah Interval Rekaman alternatif 1
Prosedur	1. Mengeklik tombol “Set Interval”. 2. Mengosongkan dialog interval. 3. Mengeklik tombol “Apply”.
Expected Result	Sistem menampilkan pesan peringatan “Interval must higher than 0”. Nilai interval rekaman tidak dirubah.
Result	<i>As expected</i>
Status	Valid

Tabel 6.37 Kasus uji dan hasil uji Mengubah Interval Rekaman (*valid input boundary value analysis*)

Kode Kebutuhan	LUPR-F-02
Nama Kasus Uji	Mengubah Interval Rekaman
Prosedur	1. Mengeklik tombol “Set Interval”. 2. Mengisi nilai 1 pada dialog interval. 3. Mengeklik tombol “Apply”.
Expected Result	Sistem mengubah nilai interval rekaman dan menampilkan pesan notifikasi “Interval changed to 1”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.38 Kasus uji dan hasil uji Mengubah Interval Rekaman alternatif 1 (*invalid input boundary value analysis*)

Kode Kebutuhan	LUPR-F-02
Nama Kasus Uji	Mengubah Interval Rekaman alternatif 1
Prosedur	1. Mengeklik tombol “Set Interval”. 2. Mengisi nilai 0 pada dialog interval. 3. Mengeklik tombol “Apply”.
Expected Result	Sistem menampilkan pesan peringatan “Interval must higher than 0”. Nilai interval rekaman tidak dirubah.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Mengubah Interval Rekaman” yang memiliki 2 kasus uji dari teknik *equivalence partitioning* pada Tabel 6.35 dan Tabel 6.36 serta 2 kasus uji dengan teknik *boundary value analysis* pada Tabel 6.37 dan Tabel 6.38 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Mengubah Interval Rekaman” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.3 Pengujian Validasi Melihat Seluruh Daftar Rekaman

Pengujian validasi “Melihat Seluruh Daftar Rekaman” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-01. Terdapat tiga kasus uji pada pengujian ini. Kasus uji pertama pada Tabel 6.39 memastikan bahwa sistem menampilkan seluruh daftar rekaman mahasiswa, Tabel 6.40 memastikan bahwa sistem menu-

tup dialog pemilihan rekaman, dan kasus uji ketiga pada Tabel 6.41 memastikan sistem menampilkan pesan peringatan bahwa direktori berkas tidak valid.

Tabel 6.39 Kasus uji dan hasil uji Melihat Seluruh Daftar Rekaman

Kode Kebutuhan	LUPV-F-01
Nama Kasus Uji	Melihat Seluruh Daftar Rekaman
Prosedur	<ol style="list-style-type: none"> 1. Mengklik tombol “Open Records”. 2. Memilih direktori rekaman. 3. Mengklik tombol “Choose”.
Expected Result	Sistem menampilkan seluruh daftar rekaman mahasiswa, dengan data yang ditampilkan meliputi nama, nim, total rekaman, waktu rekaman awal, waktu rekaman akhir dan durasi pengerajan.
Result	<i>As expected</i>
Status	Valid

Tabel 6.40 Kasus uji dan hasil uji Melihat Seluruh Daftar Rekaman alternatif 1

Kode Kebutuhan	LUPV-F-01
Nama Kasus Uji	Melihat Seluruh Daftar Rekaman alternatif 1
Prosedur	<ol style="list-style-type: none"> 1. Mengklik tombol “Open Records”. 2. Memilih direktori rekaman. 3. Mengklik tombol “Cancel”.
Expected Result	Sistem menutup dialog pemilihan rekaman dan berkas rekaman tidak dibuka.
Result	<i>As expected</i>
Status	Valid

Tabel 6.41 Kasus uji dan hasil uji Melihat Seluruh Daftar Rekaman alternatif 2

Kode Kebutuhan	LUPV-F-01
Nama Kasus Uji	Melihat Seluruh Daftar Rekaman alternatif 2
Prosedur	<ol style="list-style-type: none"> 1. Mengklik tombol “Open Records”. 2. Memilih direktori rekaman yang tidak valid, yaitu tidak sesuai dengan spesifikasi kebutuhan kode LUPV-F-01-02 dan LUPV-F-01-03. 3. Mengklik tombol “Choose”.

Expected Result	Sistem menampilkan pesan peringatan “ <i>Not a valid Task directory (baris baru) Contains invalid Task: «direktori yang tidak valid»</i> ” dan berkas rekaman tidak dibuka.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Melihat Seluruh Daftar Rekaman” yang memiliki 3 kasus uji yaitu pada Tabel 6.39, Tabel 6.40, dan Tabel 6.41 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Melihat Seluruh Daftar Rekaman” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.4 Pengujian Validasi Melihat Hasil Rekaman

Pengujian validasi “Melihat Hasil Rekaman” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-02. Terdapat empat kasus uji pada pengujian ini. Kasus uji pertama pada Tabel 6.42 memastikan bahwa sistem menampilkan hasil rekaman, Tabel 6.43 memastikan bahwa sistem menampilkan pesan informasi bahwa tidak ada berkas yang dipilih, kasus uji ketiga pada Tabel 6.44 memastikan bahwa sistem menampilkan pesan informasi bahwa tidak ada rekaman pada suatu berkas, dan kasus uji keempat pada Tabel 6.45 memastikan bahwa sistem menampilkan pesan peringatan untuk memilih berkas tugas.

Tabel 6.42 Kasus uji dan hasil uji Melihat Hasil Rekaman

Kode Kebutuhan	LUPV-F-02
Nama Kasus Uji	Melihat Hasil Rekaman
Prosedur	<ol style="list-style-type: none"> 1. Mengklik <i>clickable</i> baris mahasiswa pada seluruh daftar rekaman. 2. Mengklik <i>clickable</i> baris daftar rekaman. 3. Memilih nama berkas pada tombol <i>combo box</i> “<i>Filename</i>”. 4. Memilih tombol “<i>Line Insertion and Deletion</i>”. 5. Memilih tombol “<i>Show Mode</i>” dan “<i>Diff Mode</i>”.

Expected Result	Sistem Menampilkan hasil rekaman dan mengalihkan tampilan berkas antar mode “show” dan “diff”. Hasil rekaman yang ditampilkan terdiri dari tanggal dan waktu rekaman, isi perubahan berkas, jumlah baris yang dihapus dan ditambah pada berkas, alamat IP, seluruh window, window yang aktif, nama login, dan nama piranti.
Result	<i>As expected</i>
Status	Valid

Tabel 6.43 Kasus uji dan hasil uji Melihat Hasil Rekaman alternatif 1

Kode Kebutuhan	LUPV-F-02
Nama Kasus Uji	Melihat Hasil Rekaman alternatif 1
Prosedur	<ol style="list-style-type: none"> 1. Mengklik <i>clickable</i> baris mahasiswa pada seluruh daftar rekaman. 2. Mengklik <i>clickable</i> baris daftar rekaman.
Expected Result	Sistem menampilkan pesan informasi pada berkas rekaman dengan pesan “ <i>No file selected, please select one</i> ”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.44 Kasus uji dan hasil uji Melihat Hasil Rekaman alternatif 2

Kode Kebutuhan	LUPV-F-02
Nama Kasus Uji	Melihat Hasil Rekaman alternatif 2
Prosedur	<ol style="list-style-type: none"> 1. Mengklik <i>clickable</i> baris mahasiswa pada seluruh daftar rekaman. 2. Mengklik <i>clickable</i> baris daftar rekaman. 3. Memilih nama berkas pada tombol <i>combo box</i> “<i>Filenam</i>e”.
Expected Result	Sistem menampilkan pesan informasi pada berkas rekaman dengan pesan “ <i>No available record for «nama berkas» in this period</i> ”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.45 Kasus uji dan hasil uji Melihat Hasil Rekaman alternatif 3

Kode Kebutuhan	LUPV-F-02
Nama Kasus Uji	Melihat Hasil Rekaman alternatif 3
Prosedur	1. Mengklik <i>clickable</i> baris mahasiswa pada seluruh daftar rekaman. 2. Mengklik <i>clickable</i> baris daftar rekaman. 3. Memilih tombol “ <i>Line Insertion and Deletion</i> ”.
Expected Result	Sistem menampilkan pesan peringatan “ <i>Please choose a file</i> ”.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Melihat Hasil Rekaman” yang memiliki 4 kasus uji yaitu pada Tabel 6.42, Tabel 6.43, Tabel 6.44 dan Tabel 6.45 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Melihat Hasil Rekaman” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.5 Pengujian Validasi Mencari Tersangka

Pengujian validasi “Mencari Tersangka” membutuhkan *range* nilai yang harus dimasukkan penguji untuk melakukan validasi. Agar terhindar dari *rigorous testing*, digunakan teknik *boundary value analysis* dan *equivalence partitioning* untuk menentukan nilai *input* yang dimasukkan dalam pengujian. Nilai yang valid adalah semua bilangan bulat di atas 0.

Kasus uji yang didapatkan dengan teknik *equivalence partitioning*:

- Kasus uji *valid input*: “2”
- Kasus uji *invalid input*: “kolom kosong”

Kasus uji yang didapatkan dengan teknik *boundary value analysis*:

- Kasus uji *valid input*: “1”
- Kasus uji *invalid input*: “0”

Terdapat dua kasus uji dengan teknik *equivalence partitioning*, pertama pada Tabel 6.46 dengan nilai *input* “2” memastikan bahwa sistem menampilkan hasil

pencarian, kasus uji kedua pada Tabel 6.47 dengan mengosongkan kolom memastikan bahwa sistem menampilkan pesan peringatan. Teknik *boundary value analysis* juga menghasilkan dua kasus uji. Pertama pada Tabel 6.48 dengan nilai *input* “1” memastikan bahwa sistem menampilkan hasil pencarian, kasus uji kedua pada Tabel 6.49 dengan nilai *input* “0” memastikan bahwa sistem menampilkan pesan peringatan.

Dua kasus uji lainnya terdapat pada Tabel 6.50 untuk memastikan bahwa sistem menampilkan informasi jika tidak ada *suspect* yang ditemukan, dan pada Tabel 6.51 untuk memastikan bahwa sistem menampilkan pesan peringatan jika tidak ada berkas yang dipilih.

Tabel 6.46 Kasus uji dan hasil uji Mencari Tersangka (*valid input equivalence partitioning*)

Kode Kebutuhan	LUPV-F-03
Nama Kasus Uji	Mencari Tersangka
Prosedur	1. Memilih nama berkas dari tombol <i>combo box</i> “ <i>Filenam</i> e”. 2. Mengisi nilai 2 pada kolom “ <i>Insertion Limit</i> ”. 3. Mengeklik tombol “ <i>Search</i> ”.
Expected Result	Sistem menampilkan hasil pencarian tersangka.
Result	<i>As expected</i>
Status	Valid

Tabel 6.47 Kasus uji dan hasil uji Mencari Tersangka alternatif 1 (*invalid input equivalence partitioning*)

Kode Kebutuhan	LUPV-F-03
Nama Kasus Uji	Mencari Tersangka alternatif 2
Prosedur	1. Memilih nama berkas dari tombol <i>combo box</i> “ <i>Filenam</i> e”. 2. Mengosongkan kolom “ <i>Insertion Limit</i> ”. 3. Mengeklik tombol “ <i>Search</i> ”.
Expected Result	Sistem menampilkan pesan peringatan “ <i>Please set limit higher than 0. Above 10 is recommended</i> ”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.48 Kasus uji dan hasil uji Mencari Tersangka (*valid input boundary value analysis*)

Kode Kebutuhan	LUPV-F-03
Nama Kasus Uji	Mencari Tersangka
Prosedur	1. Memilih nama berkas dari tombol <i>combo box</i> “ <i>Filename</i> ”. 2. Mengisi nilai 1 pada kolom “ <i>Insertion Limit</i> ”. 3. Mengklik tombol “ <i>Search</i> ”.
Expected Result	Sistem menampilkan hasil pencarian tersangka.
Result	<i>As expected</i>
Status	Valid

Tabel 6.49 Kasus uji dan hasil uji Mencari Tersangka alternatif 1 (*invalid input boundary value analysis*)

Kode Kebutuhan	LUPV-F-03
Nama Kasus Uji	Mencari Tersangka alternatif 2
Prosedur	1. Memilih nama berkas dari tombol <i>combo box</i> “ <i>Filename</i> ”. 2. Mengisi nilai 0 pada kolom “ <i>Insertion Limit</i> ”. 3. Mengklik tombol “ <i>Search</i> ”.
Expected Result	Sistem menampilkan pesan peringatan “ <i>Please set limit higher than 0. Above 10 is recommended</i> ”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.50 Kasus uji dan hasil uji Mencari Tersangka alternatif 2

Kode Kebutuhan	LUPV-F-03
Nama Kasus Uji	Mencari Tersangka alternatif 2
Prosedur	1. Memilih nama berkas dari tombol <i>combo box</i> “ <i>Filename</i> ”. 2. Mengisi nilai 1000 pada kolom “ <i>Insertion Limit</i> ”. 3. Mengklik tombol “ <i>Search</i> ”.
Expected Result	Sistem menampilkan pesan informasi “ <i>No suspect found</i> ”.

Result	<i>As expected</i>
Status	Valid

Tabel 6.51 Kasus uji dan hasil uji Mencari Tersangka alternatif 3

Kode Kebutuhan	LUPV-F-03
Nama Kasus Uji	Mencari Tersangka alternatif 3
Prosedur	1. Mengisi nilai bilangan bulat di atas 0 pada kolom “ <i>Insertion Limit</i> ”. 2. Mengklik tombol “ <i>Search</i> ”.
Expected Result	Sistem menampilkan pesan peringatan “ <i>Please select a file</i> ”.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Mencari tersangka” memiliki 6 kasus uji. Keenam kasus uji pada Tabel 6.46, Tabel 6.47, Tabel 6.48, Tabel 6.49, Tabel 6.50, dan Tabel 6.51 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Mencari tersangka” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.6 Pengujian Validasi Mencari *Window*

Pengujian validasi “Mencari *Window*” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-04. Terdapat tiga kasus uji pada pengujian ini. Kasus uji pertama pada Tabel 6.52 memastikan bahwa sistem menampilkan hasil pencarian *window*, kasus uji kedua pada Tabel 6.53 memastikan bahwa sistem menampilkan pesan informasi bahwa pencarian *window* tidak ditemukan, dan kasus uji ketiga pada Tabel 6.54 memastikan sistem menampilkan pesan peringatan bahwa kolom pencarian *window* kosong.

Tabel 6.52 Kasus uji dan hasil uji Mencari *Window*

Kode Kebutuhan	LUPV-F-04
Nama Kasus Uji	Mencari <i>Window</i>
Prosedur	1. Mengisi nama window yang ingin dicari pada kolom “ <i>Window Name</i> ”. 2. Mengklik tombol “ <i>Search</i> ”.
Expected Result	Sistem menampilkan hasil pencarian <i>window</i> .

Result	<i>As expected</i>
Status	Valid

Tabel 6.53 Kasus uji dan hasil uji Mencari Window alternatif 1

Kode Kebutuhan	LUPV-F-04
Nama Kasus Uji	Mencari Window alternatif 1
Prosedur	1. Mengisi “11111111” pada kolom “Window Name”. 2. Mengeklik tombol “Search”.
Expected Result	Sistem menampilkan pesan informasi “No window name for “11111111” found”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.54 Kasus uji dan hasil uji Mencari Window alternatif 2

Kode Kebutuhan	LUPV-F-04
Nama Kasus Uji	Mencari Window alternatif 2
Prosedur	1. Mengeklik tombol “Search”.
Expected Result	Sistem menampilkan pesan peringatan “Please supply the window name”.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Mencari Window” yang memiliki 3 kasus uji yaitu pada Tabel 6.52, Tabel 6.53, dan Tabel 6.54 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Mencari Window” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.7 Pengujian Validasi Melihat Seluruh Alamat IP

Pengujian validasi “Melihat Seluruh Alamat IP” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-06. Terdapat satu kasus uji pada pengujian ini, yaitu kasus uji pada Tabel 6.55 memastikan bahwa sistem menampilkan seluruh alamat IP.

Tabel 6.55 Kasus uji dan hasil uji Melihat Seluruh Alamat IP

Kode Kebutuhan	LUPV-F-05
Nama Kasus Uji	Melihat Seluruh Alamat IP
Prosedur	1. Mengklik tombol “Show all IP addresses”.
Expected Result	Sistem menampilkan seluruh alamat IP mahasiswa.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Melihat Seluruh Alamat IP” yang memiliki 1 kasus uji yaitu pada Tabel 6.55 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Melihat Seluruh Alamat IP” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.8 Pengujian Validasi Melihat Grafik *Edit-distance*

Pengujian validasi “Melihat Grafik *Edit-distance*” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-06. Terdapat dua kasus uji pada pengujian ini. Kasus uji pertama pada Tabel 6.56 memastikan bahwa sistem menampilkan grafik *edit-distance* dari tugas mahasiswa, kasus uji kedua pada Tabel 6.57 memastikan bahwa sistem menampilkan pesan peringatan bahwa salah satu tombol *combo box* nama mahasiswa dan berkas tugas belum dipilih.

Tabel 6.56 Kasus uji dan hasil uji Melihat Grafik *Edit-distance*

Kode Kebutuhan	LUPV-F-06
Nama Kasus Uji	Melihat Grafik <i>Edit-distance</i>
Prosedur	1. Memilih nama mahasiswa dari tombol <i>combo box</i> “Current Student”. 2. Memilih nama berkas dari tombol <i>combo box</i> “Filename”. 3. Mengklik tombol “Show Mode”.
Expected Result	Sistem menampilkan grafik <i>edit-distance</i> dari tugas mahasiswa.
Result	<i>As expected</i>
Status	Valid

Tabel 6.57 Kasus uji dan hasil uji Melihat Grafik *Edit-distance* alternatif 1

Kode Kebutuhan	LUPV-F-06
Nama Kasus Uji	Melihat Grafik <i>Edit-distance</i> alternatif 1
Prosedur	1. Mengklik tombol “Show Mode”.
Expected Result	Sistem menampilkan pesan peringatan “Please fill one of the pairs”.
Result	As expected
Status	Valid

Hasil pengujian validasi “Melihat Grafik *Edit-distance*” yang memiliki 2 kasus uji yaitu pada Tabel 6.56, dan Tabel 6.57 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Melihat Grafik *Edit-distance*” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.9 Pengujian Validasi Melihat Grafik *Edit-distance* Tugas Sebelumnya

Pengujian validasi “Melihat Grafik *Edit-distance* Tugas Sebelumnya” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-07. Terdapat empat kasus uji pada pengujian ini. Kasus uji pertama pada Tabel 6.58 memastikan bahwa sistem menampilkan grafik *edit-distance* dari tugas sebelumnya, kasus uji kedua pada Tabel 6.59 memastikan bahwa sistem menutup dialog pencarian berkas *edit-distance*, kasus uji ketiga pada Tabel 6.60 memastikan bahwa sistem menampilkan pesan peringatan bahwa berkas *edit-distance* belum dibuka, dan kasus uji keempat pada Tabel 6.61 memastikan bahwa salah satu tombol *combo box* nama mahasiswa atau berkas tugas belum dipilih.

Tabel 6.58 Kasus uji dan hasil uji Melihat Grafik *Edit-distance* Tugas Sebelumnya

Kode Kebutuhan	LUPV-F-07
Nama Kasus Uji	Melihat Grafik <i>Edit-distance</i> Tugas Sebelumnya

Prosedur	<ol style="list-style-type: none"> 1. Mengeklik tombol “Load Edit-distance File”. 2. Memilih berkas <i>edit-distance</i> tugas sebelumnya pada dialog pencarian berkas. 3. Mengeklik tombol “Open”. 4. Memilih nama mahasiswa dari tombol <i>combo box</i> “Previous Student”. 5. Memilih nama berkas dari tombol <i>combo box</i> “Filename”. 6. Mengeklik tombol “Show Mode”.
Expected Result	Sistem menampilkan grafik <i>edit-distance</i> tugas sebelumnya.
Result	<i>As expected</i>
Status	Valid

Tabel 6.59 Kasus uji dan hasil uji Melihat Grafik *Edit-distance* Tugas Sebelumnya alternatif 1

Kode Kebutuhan	LUPV-F-07
Nama Kasus Uji	Melihat Grafik <i>Edit-distance</i> Tugas Sebelumnya alternatif 1.
Prosedur	<ol style="list-style-type: none"> 1. Mengeklik tombol “Load Edit-distance File”. 2. Memilih berkas <i>edit-distance</i> tugas sebelumnya pada dialog pencarian berkas. 3. Mengeklik tombol “Cancel”.
Expected Result	Sistem menutup dialog pencarian berkas <i>edit-distance</i> tugas sebelumnya.
Result	<i>As expected</i>
Status	Valid

Tabel 6.60 Kasus uji dan hasil uji Melihat Grafik *Edit-distance* Tugas Sebelumnya alternatif 2

Kode Kebutuhan	LUPV-F-07
Nama Kasus Uji	Melihat Grafik <i>Edit-distance</i> Tugas Sebelumnya alternatif 2

Prosedur	1. Memilih nama mahasiswa dari tombol <i>combo box</i> “ <i>Previous Student</i> ”. 2. Mengklik tombol “ <i>Show Mode</i> ”.
Expected Result	Sistem menampilkan pesan peringatan “ <i>Please load edit-distance file first</i> ”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.61 Kasus uji dan hasil uji Melihat Grafik *Edit-distance* Tugas Sebelumnya alternatif 3

Kode Kebutuhan	LUPV-F-07
Nama Kasus Uji	Melihat Grafik <i>Edit-distance</i> Tugas Sebelumnya alternatif 3.
Prosedur	1. Mengklik tombol “ <i>Load Edit-distance File</i> ”. 2. Memilih berkas <i>edit-distance</i> tugas sebelumnya pada dialog pencarian berkas. 3. Mengklik tombol “ <i>Open</i> ”. 4. Mengklik tombol “ <i>Show Mode</i> ”.
Expected Result	Sistem menampilkan pesan peringatan “ <i>Please fill one of the pairs</i> ”.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Melihat Grafik *Edit-distance* Tugas Sebelumnya” yang memiliki 4 kasus uji yaitu pada Tabel 6.58, Tabel 6.59, Tabel 6.60, dan Tabel 6.61 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Melihat Grafik *Edit-distance* Tugas Sebelumnya” sudah sesuai dengan kebutuhan yang definisikan.

6.3.10 Pengujian Validasi Menyimpan Grafik *Edit-distance*

Pengujian validasi “Menyimpan Grafik *Edit-distance*” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-08. Terdapat satu kasus uji pada pengujian ini, yaitu kasus uji pada Tabel 6.62 memastikan bahwa sistem menyimpan grafik *edit-distance*.

Tabel 6.62 Kasus uji dan hasil uji Menyimpan Grafik *Edit-distance*

Kode Kebutuhan	LUPV-F-08
Nama Kasus Uji	Menyimpan Grafik <i>Edit-distance</i>
Prosedur	1. Mengklik tombol “Save Graph”.
Expected Result	Sistem menyimpan grafik <i>edit-distance</i> dengan nama “«namamahasiswa-nim».png” dan menampilkan pesan informasi “Graph saved to «lokasi berkas»” menandakan bahwa berkas berhasil disimpan.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Menyimpan Grafik *Edit-distance*” yang memiliki 1 kasus uji yaitu pada Tabel 6.62 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Menyimpan Grafik *Edit-distance*” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.11 Pengujian Validasi Mengekspor Semua Nilai *Edit-distance*

Pengujian validasi “Mengekspor Semua Nilai *Edit-distance*” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-09. Terdapat dua kasus uji pada pengujian ini. Kasus uji pertama pada Tabel 6.63 memastikan bahwa sistem mengekspor semua nilai *edit-distance*, dan kasus uji kedua pada Tabel 6.64 memastikan bahwa sistem menutup dialog pemilihan berkas dan nilai *edit-distance* tidak dieksport.

Tabel 6.63 Kasus uji dan hasil uji Mengekspor Semua Nilai *Edit-distance*

Kode Kebutuhan	LUPV-F-09
Nama Kasus Uji	Mengekspor Semua Nilai <i>Edit-distance</i>
Prosedur	1. Mengklik tombol “Export <i>Edit-distance</i> ”. 2. Mengklik tombol “Export”.
Expected Result	Sistem mengekspor semua nilai <i>edit-distance</i> pada berkas dengan nama “«namatugas»-editdistance.lup”, dan menampilkan pesan informasi “ <i>Edit-distance exported to «lokasi berkas»</i> ”.
Result	<i>As expected</i>
Status	Valid

Tabel 6.64 Kasus uji dan hasil uji Mengekspor Semua Nilai *Edit-distance* alternatif 1

Kode Kebutuhan	LUPV-F-09
Nama Kasus Uji	Mengekspor Semua Nilai <i>Edit-distance</i> alternatif 1
Prosedur	1. Mengeklik tombol “Export <i>Edit-distance</i> ”. 2. Mengeklik tombol “Cancel”.
Expected Result	Sistem menutup dialog pemilihan berkas dan nilai <i>edit-distance</i> tidak diekspor.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Mengekspor Semua Nilai *Edit-distance*” yang memiliki 2 kasus uji yaitu pada Tabel 6.63, dan Tabel 6.64 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Mengekspor Semua Nilai *Edit-distance*” sudah sesuai dengan kebutuhan yang didefinisikan.

6.3.12 Pengujian Validasi Mengalihkan Format Waktu

Pengujian validasi “Mengalihkan Format Waktu” dilakukan untuk menguji spesifikasi kebutuhan fungsional LUPV-F-10. Terdapat satu kasus uji pada pengujian ini, yaitu kasus uji pada Tabel 6.65 memastikan bahwa sistem mengalihkan format waktu.

Tabel 6.65 Kasus uji dan hasil uji Mengalihkan Format Waktu

Kode Kebutuhan	LUPV-F-10
Nama Kasus Uji	Mengalihkan Format Waktu
Prosedur	1. Mengeklik tombol “Real DateTime” dan “Relative DateTime”.
Expected Result	Sistem mengalihkan format waktu.
Result	<i>As expected</i>
Status	Valid

Hasil pengujian validasi “Mengalihkan Format Waktu” yang memiliki 1 kasus uji yaitu pada Tabel 6.65 menghasilkan nilai valid. Maka dapat dipastikan fungsionalitas “Mengalihkan Format Waktu” sudah sesuai dengan kebutuhan yang didefinisikan.

6.4 Automated Testing

Automated testing digunakan untuk mempercepat proses pengujian. Pada tahapan ini dibangun *script* yang menjalankan kasus uji secara otomatis. Kasus uji didapatkan dari pengujian unit dan pengujian integrasi yang telah dilakukan pada tahapan sebelumnya. *Script* dibangun menggunakan kakas bantu *Pytest*, dan dijalankan secara otomatis dengan kakas bantu *Gitlab-CI*. Terdapat tiga sampel yang dipaparkan, yaitu *script automated testing* untuk *function get_all_windows* pada *class Controller*, *function populate_log* pada *class LogController*, dan *function construct_ed_graph_path* pada *class SearchController*. Gambar 6.33 menunjukkan seluruh hasil *automated testing* yang berjalan dengan nilai “PASSED” tanpa ada galat.

```
tests/test_search_controller.py::TestSearchController::test_student_directories_iterator PASSED
tests/test_search_controller.py::TestSearchController::test_record_iterator PASSED
tests/test_search_controller.py::TestSearchController::test_analyze_suspects PASSED
tests/test_search_controller.py::TestSearchController::test_group_by_name PASSED
tests/test_search_controller.py::TestSearchController::test_get_suspects PASSED
tests/test_search_controller.py::TestSearchController::test_get_student_ips PASSED
tests/test_search_controller.py::TestSearchController::test_group_by_ip PASSED
tests/test_search_controller.py::TestSearchController::test_multigroup_child PASSED
tests/test_search_controller.py::TestSearchController::test_get_student_ip_groups PASSED
tests/test_search_controller.py::TestSearchController::test_idx_of_substring PASSED
tests/test_search_controller.py::TestSearchController::test_collect_student_windows PASSED
tests/test_search_controller.py::TestSearchController::test_get_student_windows PASSED
tests/test_search_controller.py::TestSearchController::test_populate_student_dirs PASSED
tests/test_search_controller.py::TestSearchController::test_load_prev_edidistances PASSED
tests/test_search_controller.py::TestSearchController::test_get_prev_student_names PASSED
tests/test_search_controller.py::TestSearchController::test_get_prev_filename_sample PASSED
tests/test_search_controller.py::TestSearchController::test_calc_prev_edidistances PASSED
tests/test_search_controller.py::TestSearchController::test__get_student_records PASSED
tests/test_search_controller.py::TestSearchController::test_calc_edidistances PASSED
tests/test_search_controller.py::TestSearchController::test_create_lupvnotes_dir PASSED
tests/test_search_controller.py::TestSearchController::test_construct_ed_graph_path PASSED
tests/test_search_controller.py::TestSearchController::test_construct_editistance_path PASSED
tests/test_search_controller.py::TestSearchController::test_export_edidistance PASSED
=====
===== 91 passed in 8.62 seconds =====
```

Gambar 6.33 Seluruh hasil *automated testing*

6.4.1 Test Script Automated Testing Class Controller Function

get_all_windows

Tabel Kode 6.15 memaparkan *script* yang digunakan untuk *automated testing* pada *function get_all_windows*. Pengujian dilakukan secara terisolasi sehingga dilakukan *stub* pada proses *Popen*. *Driver* kemudian memanggil *function get_all_windows* dan melakukan *assert* pada nilai yang dikembalikan.

No	get_all_windows_with_fake
1	<code>def test_get_all_windows_with_fake(self, ctrl, monkeypatch):</code>
2	<code> """Test get_all_windows with Fake Object."""</code>

```

3     window_title =
4         → b"0x006000ab 0 machine-name foo_window_title"
5
6     def fake_communicate(input=None, timeout=None):
7         return window_title, "err"
8
9     Lupr.controllers.controller.Popen = FakePopen
10    Lupr.controllers.controller.Popen.communicate =
11        → fake_communicate
12
13    output = ctrl.get_all_windows()
14    assert output == "foo_window_title\n"

```

Tabel Kode 6.15 Test script function get_all_windows_with_fake

Hasil pengujian *function get_all_windows* dengan menggunakan *script automated testing* pada Tabel Kode 6.15 menghasilkan nilai “PASSED” seperti yang terlihat pada Gambar 6.33. Maka dapat dipastikan penggunaan *script automated testing* dapat mempercepat proses pengujian.

6.4.2 *Test Script Automated Testing Class LogController Function populate_logs*

Tabel Kode 6.16 memaparkan *script* yang digunakan untuk *automated testing* pada *function populate_logs*. Pengujian dilakukan secara terisolasi sehingga dilakukan *stub* pada *function is_exist* dan *function student_records*. Driver kemudian memanggil *function populate_logs* dan melakukan *assert* pada nilai yang dikembalikan.

No	populate_logs
1	<pre> def test_populate_logs_no_records(self, log_ctrl_model): """Test populating student logs using dummy data.""" log_ctrl, log_model = log_ctrl_model log_model.student_records = [] log_model.is_exists = cf.fake_is_exists ani_logs = list(log_ctrl.populate_logs(selected_file="tugas-tif.txt" </pre>

10))
11	
12	assert ani_logs == []

Tabel Kode 6.16 Test script function populate_logs

Hasil pengujian *function populate_logs* dengan menggunakan *script automated testing* pada Tabel Kode 6.16 menghasilkan nilai “PASSED” seperti yang terlihat pada Gambar 6.33. Maka dapat dipastikan penggunaan *script automated testing* dapat mempercepat proses pengujian.

6.4.3 *Test Script Automated Testing Class SearchController Function construct_ed_graph_path*

Tabel Kode 6.17 memaparkan *script* yang digunakan untuk *automated testing* pada *function construct_ed_graph_path*. *Function construct_ed_graph_path* tidak memanggil *function* lain sehingga tidak diperlukan adanya *stub*. *Driver* kemudian memanggil *function construct_ed_graph_path* dan melakukan *assert* pada nilai yang dikembalikan.

No	construct_ed_graph_path
1	def test_construct_ed_graph_path(self, search_ctrl): <comment>"""Test constructing editdistance graph path."""</comment> graph_path = search_ctrl.construct_ed_graph_path("ani-1111") graph_path_2 = search_ctrl.construct_ed_graph_path("ani-1111", "budi-2222") assert "/lupv-notes/ani-1111.png" in graph_path assert "/lupv-notes/ani-1111_budi-2222.png" in graph_path_2

Tabel Kode 6.17 Test script function construct_ed_graph_path

Hasil pengujian *function construct_ed_graph_path* dengan menggunakan *script automated testing* pada Tabel Kode 6.17 menghasilkan nilai “PASSED” seperti yang terlihat pada Gambar 6.33. Maka dapat dipastikan penggunaan *script automated testing* dapat mempercepat proses pengujian.

6.5 Pengujian *Compatibility*

Pengujian *compatibility* dilakukan untuk memastikan aplikasi yang dibangun dapat berjalan dengan baik dan dapat menangani perbedaan format nama *window* pada enam *desktop environment* utama yang ada pada sistem operasi *GNU/Linux*. Enam *desktop environment* utama tersebut adalah *GNOME*, *KDE*, *XFCE*, *Unity*, *Cinnamon* dan *Phanteon*. Pengujian *compatibility* pada sistem *Lup Recorder* dilakukan dengan menjalankan sistem pada seluruh *desktop environment* yang tertera dalam Tabel 6.66, pengujian dilakukan oleh penguji yang bersangkutan dalam tabel yang sama. Terlihat pada Tabel 6.66 sistem juga diujikan terhadap dua *desktop environment* tambahan yaitu *OpenBox* dan *i3wm*. Gambar 6.34 menunjukkan seluruh hasil rekaman sistem *Lup Recorder* yang dapat dibaca oleh sistem *Lup Viewer*. Hal ini menandakan bahwa sistem *Lup Viewer* dapat berjalan dengan baik pada seluruh *desktop environment* dalam Tabel 6.66. Pengujian sistem *Lup Viewer* dilakukan dengan menjalankannya pada *desktop environment* yang berbeda-beda. Gambar 6.35 menunjukkan sistem *Lup Viewer* dapat berjalan dengan baik pada *desktop environment* *i3wm* dan Gambar 6.36 menunjukkan sistem *Lup Viewer* berjalan dengan baik pada *desktop environment* *XFCE*.

Tabel 6.66 Daftar *desktop environment* yang diuji

No	Nama Penguji	Desktop Environment	Versi
1	Bintang Dimas PAR	<i>GNOME</i>	3.28.2
2	Muhammad Iqbal K	<i>GNOME</i>	3.28.2
3	Laode Muhamad Fauzan	<i>KDE</i>	5.9.15
4	Husein Abdulbar	<i>KDE</i>	5.15.5
5	Rofy Firmansyah R	<i>KDE</i>	5.15.5
6	Satria Adhi Kharisma	<i>XFCE</i>	4.12
7	Dese Narfa Firmansyah	<i>XFCE</i>	4.12
8	Rizaldy Firmansyah Y	<i>Unity</i>	7
9	Insan Nurzaman	<i>Cinnamon</i>	3.6.6
10	Jerna Ferda Kusuma	<i>Cinnamon</i>	4.0.10
11	Rahmat Guntur Husodo	<i>Cinnamon</i>	4.0.10

No	Nama Penguji	Desktop Environment	Versi
12	Febristya AF	<i>Pantheon</i>	5.0
13	Achmad Rizki Aditama	<i>OpenBox</i>	3.6.1
14	Mohammad Anton RS	<i>i3wm</i>	4.16.1

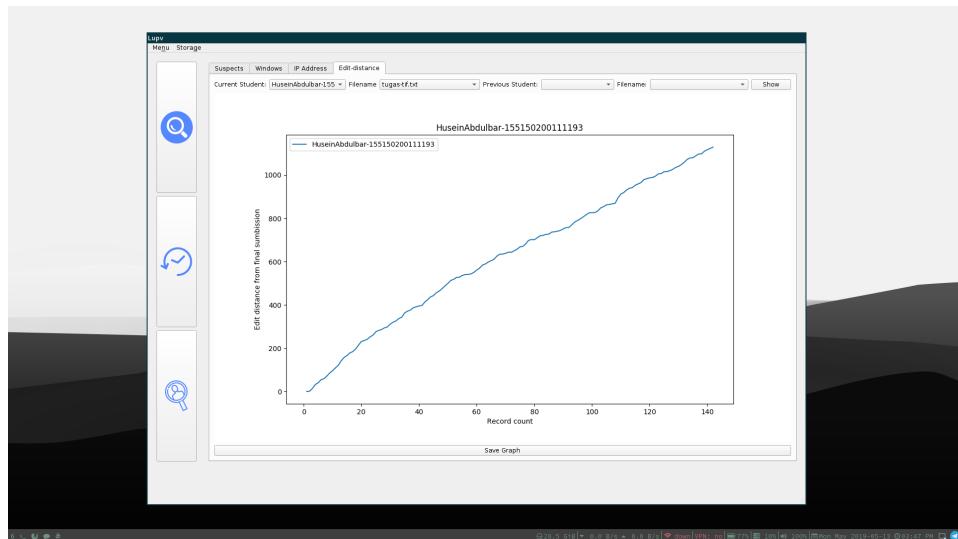
Menu Storage

Relative DateTime Real DateTime

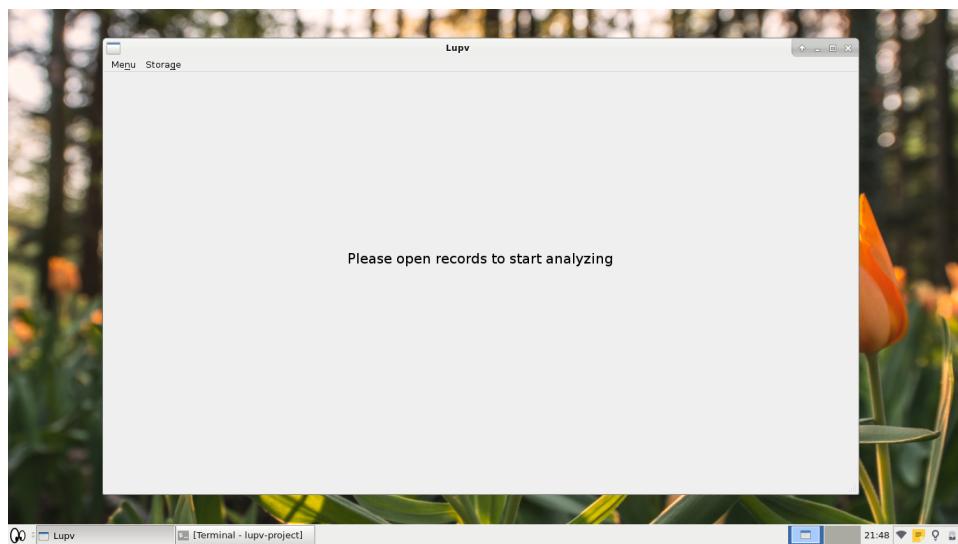
Appearance Storage

	Name	Student ID	Total Records	First Record	Last Record	Work Duration
1	SatriaAdhiKharisma	155150207111119	193	Thu, 21 Mar 2019, 20:37:00	Thu, 21 Mar 2019, 20:49:54	0:12:54
2	RizaldyFirmansyahYulianto	155150207111112	273	Fri, 22 Mar 2019, 01:27:01	Fri, 22 Mar 2019, 03:42:27	2:15:26
3	DeseNarfaFirmansyah	155150201111153	527	Thu, 21 Mar 2019, 18:44:43	Thu, 21 Mar 2019, 19:23:30	0:38:47
4	LaodemMuhamadFauzan	155150200111158	146	Wed, 20 Mar 2019, 16:35:02	Wed, 20 Mar 2019, 17:06:15	0:31:13
5	InsanNurzaman	16515020011033	300	Wed, 20 Mar 2019, 19:38:20	Wed, 20 Mar 2019, 20:03:24	0:25:04
6	AchmadRizkiAditama	155150207111158	344	Wed, 20 Mar 2019, 20:58:02	Wed, 20 Mar 2019, 21:21:33	0:23:31
7	BintangDimasPAR	15515020111347	373	Wed, 20 Mar 2019, 22:56:24	Wed, 20 Mar 2019, 23:35:33	0:39:09
8	FebristyaAF	15515020111126	197	Thu, 21 Mar 2019, 20:04:01	Thu, 21 Mar 2019, 20:20:24	0:16:23
9	HuseinAbdulbar	15515020011193	177	Fri, 22 Mar 2019, 08:07:48	Fri, 22 Mar 2019, 08:18:00	0:10:12
10	JernaFerdakusuma	15508020011053	148	Sat, 23 Mar 2019, 12:55:03	Sat, 23 Mar 2019, 13:15:50	0:20:47
11	MohammadAntonRizkiSyahputra	15515020011104	240	Thu, 21 Mar 2019, 20:04:03	Thu, 21 Mar 2019, 20:24:27	0:20:24
12	RofyFirmansyahR	15515020011115	270	Thu, 21 Mar 2019, 21:17:22	Thu, 21 Mar 2019, 21:32:39	0:15:17
13	RahmatGunturHusodo	16515020111282	335	Thu, 21 Mar 2019, 18:28:21	Thu, 21 Mar 2019, 18:56:18	0:27:57
14	MuhammadIqbalKurniawan	155150200111259	385	Wed, 20 Mar 2019, 19:44:08	Wed, 20 Mar 2019, 20:12:24	0:28:16

Gambar 6.34 Sistem *Lup Recorder* berhasil merekam berkas tugas pada *desktop environment* yang berbeda-beda



Gambar 6.35 *Lup Viewer* pada *desktop environment i3wm*



Gambar 6.36 *Lup Viewer* pada *desktop environment XFCE*

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan dari hasil penelitian yang dilakukan adalah sebagai berikut:

1. Pada tahapan rekayasa kebutuhan didapatkan 2 aktor utama, 2 kebutuhan fungsional untuk sistem *Lup Recorder* dan 10 kebutuhan fungsional untuk sistem *Lup Viewer*. Dua aktor utama yang terlibat di dalam sistem adalah dosen dan mahasiswa. Kebutuhan utama sistem adalah merekam berkas tugas, merekam aktivitas siswa, dan memiliki sifat *platform agnostic*. Kedua sistem memiliki satu kebutuhan non-fungsional yaitu *compatibility*. Kedua sistem harus *compatible* dengan enam *desktop environment* utama pada sistem operasi *GNU/Linux*.
2. Pada tahapan perancangan didapatkan pemodelan *class diagram* yang terdiri dari 6 *class* pada sistem *Lup Recorder* dan 12 *class* pada sistem *Lup Viewer*, serta tiga sampel pemodelan *sequence diagram*. Hasil dari tahapan perancangan diimplementasikan dalam bentuk *working code* dengan bahasa pemrograman *Python*, dan antarmuka sistem diimplementasikan dengan bantuan *widget toolkit Qt*.
3. Terdapat lima pengujian yang dilakukan, yaitu pengujian unit, pengujian integrasi, pengujian validasi, *automated testing*, dan pengujian *compatibility*. Pengujian unit dilakukan dengan metode *white-box testing* dan teknik *basis path testing*, pengujian integrasi dilakukan dengan pendekatan *use-based testing* menggunakan metode *white-box testing* dan teknik *basis path testing*, pengujian validasi dilakukan dengan metode *black-box testing* dengan pemilihan nilai *input* menggunakan teknik *boundary value analysis* dan *equivalence partitioning*, *automated testing* dilakukan dengan menggunakan bantuan kertas bantu *Pytest* dan *Gitlab-CI*, dan pengujian *compatibility* dilakukan dengan menjalankan sistem pada enam *desktop environment* pada sistem operasi *GNU/Linux*. Pengujian unit dan integrasi yang dilakukan telah mencapai lebih dari 70% *code coverage* sehingga bisa dipastikan semua *path* utama selesai diuji. Pengujian validasi dilakukan terhadap seluruh *use case scenario* sehingga bisa dipastikan bahwa sistem yang dibangun memenuhi seluruh kebutuhan yang didefinisikan. *Automated testing* dijalankan tanpa menghasilkan galat sehingga pengujian dapat dilakukan dengan otomatis dan me-

mangkas waktu pengujian. Pengujian *compatibility* telah memastikan sistem berjalan dengan baik pada enam *desktop environment* berbeda.

7.2 Saran

Saran untuk penelitian selanjutnya adalah sebagai berikut:

1. Aplikasi pencegah plagiarisme dengan menggunakan metode *snapshotting* dan *user activity logging* menyimpan semua data rekaman secara lokal. Maka terdapat kemungkinan adanya manipulasi data rekaman oleh mahasiswa yang memahami struktur data rekaman sistem. Saran untuk penelitian selanjutnya adalah dengan tidak menyimpan data secara lokal, melainkan dikirim dan disimpan langsung pada penyimpanan *server*.
2. Sistem operasi yang didukung dapat ditambah dengan melakukan *porting* modul perekam pada sistem ke sistem operasi lain seperti sistem operasi *Microsoft Windows* dan *Apple Macintosh*.

DAFTAR PUSTAKA

- About draw.io* 2019. Available at: <<https://about.draw.io/about-us/>> [Accessed July 3, 2019].
- Agustina, R. & Raharjo, P., 2017. Exploring Plagiarism into Perspectives of Indonesian Academics and Students. *Journal of Education and Learning*, 11(3), pp. 262–272.
- Bell, D., 2000. *Software Engineering for Students*. Pearson Education.
- Born, A. D., 2003. Teaching tip: How to reduce plagiarism. *Journal of Information Systems Education*, 14(3), p. 223.
- Brooks Jr, F. P., 1995. *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition*, 2/E. Pearson Education India.
- Burnstein, I., 2006. *Practical software testing: a process-oriented approach*. Springer Science & Business Media.
- Capretz, L. F., 2003. A Brief History of the Object-Oriented Approach. *Software Engineering Notes*, 28(2).
- Chacon, S. & Straub, B., 2014. *Pro git*. Apress.
- DeLaRosa, A., 2018. *Log Monitoring: not the ugly sister*. Available at: <<https://blog.pandorafms.org/log-monitoring/>> [Accessed Aug. 21, 2018].
- Dijkstra, E. W., 1970. *Notes on structured programming*.
- Dweck, C. S. & Leggett, E. L., 1988. A social-cognitive approach to motivation and personality. *Psychological review*, 95(2), p. 256.
- Fowler, M., 2018. *UnitTest*. Available at: <<https://martinfowler.com/bliki/UnitTest.html>> [Accessed Sept. 23, 2018].
- Girimella, N., 2018. *Understanding and exploiting snapshot technology for data protection, Part 1: Snapshot technology overview*. Available at: <<https://www.ibm.com/developerworks/tivoli/library/t-snapsm1/>> [Accessed Aug. 21, 2018].
- Garner, H., Pulverer, B., Marusić, A., Petrovechi, M., Loadsman, J., Zhang, Y., McIntosh, I., Titus, S., Roig, M., & Anderson, M., 2012. How to stop plagiarism. *Nature*, 481(7382), pp. 21–23.
- Gregory, L., 2007. *Path Testing*.
- Hellas, A., Leinonen, J., & Ihantola, P., 2017. Plagiarism in Take-home Exams: Help-seeking, Collaboration, and Systematic Cheating. In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, pp. 238–243.

- Howard, R. M., 2002. Don't police plagiarism: just teach! *The education digest*, 67(5), p. 46.
- IEEE, 1990. *IEEE Standard Glossary of Software Engineering Terminology*.
- Kiss, A. K., 2013. Loopholes of plagiarism detection software. *Procedia-Social and Behavioral Sciences*, 106, pp. 1796–1803.
- Leung, C. H. & Cheng, S. C. L., 2017. An Instructional Approach to Practical Solutions for Plagiarism. *Universal Journal of Educational Research*, 5(9), pp. 1646–1652.
- Macbeth, S. W., Fernandez, R. L., Meyers, B. R., Tan, D. S., Robertson, G. G., Oliver, N. M., Murillo, O. E., Pedersen, E. R., Czerwinski, M. P., Spence, J. E., et al., Dec. 2007. *Logging user actions within activity context*. US Patent App. 11/426,846.
- Martins, V. T., Fonte, D., Henriques, P. R., & Cruz, D. da, 2014. Plagiarism Detection: A Tool Survey and Comparison. In: *3rd Symposium on Languages, Applications and Technologies*. Vol. 38. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 143–158.
- McLafferty, C. L. & Foust, K. M., 2004. Electronic plagiarism as a college instructor's nightmare—Prevention and detection. *Journal of Education for Business*, 79(3), pp. 186–190.
- Meuschke, N. & Gipp, B., 2013. State-of-the-art in detecting academic plagiarism. *International Journal for Educational Integrity*, 9(1).
- Modiba, P., Pieterse, V., & Haskins, B., 2016. Evaluating plagiarism detection software for introductory programming assignments. In: *Proceedings of the Computer Science Education Research Conference 2016*. ACM, pp. 37–46.
- Myers, G. J., Sandler, C., & Badgett, T., 2011. *The art of software testing*. John Wiley & Sons.
- Navarro, G., 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1), pp. 31–88.
- Neill, C. J. & Shanmuganathan, G., 2004. A Web-enabled plagiarism detection tool. *IT professional*, 6(5), pp. 19–23.
- Nshimiyiman, M., 2018. *Travis-ci - Send notification with custom message*. Available at: <<http://www.marcellin.me/blog/travis-ci-notification-custom-message/>> [Accessed Sept. 23, 2018].
- Osherove, R., 2015. *The art of unit testing*. MITP-Verlags GmbH & Co. KG.
- Park, C., 2004. Rebels without a clause: Towards an institutional framework for dealing with plagiarism by students. *Journal of further and Higher Education*, 28(3), pp. 291–306.

- Presman, R. S., 2010. *Software Engineering a Practitioner's Approach*. New York: McGraw-Hill Companies Inc.
- Pressman, R. S., 2010. *Software Engineering—A Practitioner's Approach*. Mc Graw-Hill.
- Qt About* 2018. Available at: <https://wiki.qt.io/About_Qt> [Accessed Dec. 21, 2018].
- Qt History* 2018. Available at: <https://wiki.qt.io/Qt_History> [Accessed Dec. 21, 2018].
- Rumbaugh, J., Jacobson, I., & Booch, G., 2004. *Unified modeling language reference manual, the*. Pearson Higher Education.
- Ruparelia, N. B., 2010. The history of version control. *ACM SIGSOFT Software Engineering Notes*, 35(1), pp. 5–9.
- Sanner, M. F., et al., 1999. Python: a programming language for software integration and development. *J Mol Graph Model*, 17(1), pp. 57–61.
- Savenkov, R., 2008. *How to become a software tester*. Roman Savenkov.
- Sommerville, I., 2014. *Software Engineering*. Pearson Education.
- Torvalds, L., 2018. *git mailing list*. Available at: <<https://marc.info/?l=linux-kernel&m=111314792424707>> [Accessed Aug. 21, 2018].
- Whittaker, J. A., Arbon, J., & Carollo, J., 2012. *How Google tests software*. Addison-Wesley.