

**PENGEMBANGAN *AUTO GRADER* MENGGUNAKAN METODE  
*DOMAIN DRIVEN DESIGN* DI JURUSAN ILMU KOMPUTER  
UNIVERSITAS RIAU**

**SKRIPSI**



**OLEH**  
**SAMMI ALDHI YANTO**  
**NIM.2003113948**

**PROGRAM STUDI S1 SISTEM INFORMASI  
JURUSAN ILMU KOMPUTER  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS RIAU  
PEKANBARU  
2024**

**PENGEMBANGAN *AUTO GRADER* MENGGUNAKAN METODE  
*DOMAIN DRIVEN DESIGN* DI JURUSAN ILMU KOMPUTER  
UNIVERSITAS RIAU**

**SKRIPSI**

**Diajukan sebagai Salah Satu Syarat  
Untuk Memperoleh Gelar Sarjana Komputer**



**OLEH**  
**SAMMI ALDHI YANTO**  
**NIM.2003113948**

**PROGRAM STUDI S1 SISTEM INFORMASI  
JURUSAN ILMU KOMPUTER  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS RIAU  
PEKANBARU  
2024**

**PENGEMBANGAN *AUTO GRADER* MENGGUNAKAN METODE  
DOMAIN DRIVEN DESIGN DI JURUSAN ILMU KOMPUTER  
UNIVERSITAS RIAU**

---

**Diperiksa dan Disetujui Oleh:**

**Pembimbing**

**Yanti Andriyani, S.T., M.T.I., Ph.D.**  
**NIP. 19810512 200812 2 001**

**Mengetahui**  
**Ketua Jurusan Ilmu Komputer**

**Roni Salambue, S.Kom., M.Si.**  
**NIP. 19740930 200312 1 001**

**Skripsi ini telah diuji oleh Tim Penguji Ujian Sarjana Komputer  
Program Studi S1 Sistem Informasi  
Jurusan Ilmu Komputer  
Fakultas Matematika dan Ilmu Pengetahuan Alam  
Universitas Riau Pekanbaru**

**Pada Tanggal :**

---

**Tim Penguji :**

**Ibnu Daqiqil Id, S.Kom., M.T.I., Ph.D.**                   Ketua ( )  
NIP. 19860320 201504 1 001

**Yanti Andriyani, ST., M.T.I., Ph.D.**                   Anggota ( )  
NIP. 19810512 200812 2 002

**Sonya Meitarice, M.Sc.**                           Anggota ( )  
NIP. 19890504 202203 1 005

**Mengetahui  
Dekan FMIPA Universitas Riau**

**Dr. Svamsudhuha, M.Sc.**  
NIP. 19630512 198903 1 002

## **LEMBAR PERNYATAAN**

Dengan ini penulis menyatakan bahwa skripsi yang berjudul “Pengembangan *Auto Grader* Menggunakan Metode *Domain Driven Design* di Jurusan Ilmu Komputer Universitas Riau”, benar hasil penelitian penulis dengan arahan Dosen Pembimbing dan belum pernah diajukan dalam bentuk apa pun untuk mendapatkan gelar Sarjana Komputer. Dalam skripsi ini tidak terdapat karya atau pendapat yang ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan dalam naskah dengan menyebutkan referensi yang dicantumkan dalam daftar pustaka. Pernyataan ini penulis buat dengan sesungguhnya dan apabila dikemudian hari terdapat penyimpangan dan kesalahan dalam pernyataan ini, maka penulis bersedia menerima sanksi akademik berupa pencabutan gelar yang telah penulis peroleh karena skripsi ini, serta sanksi lainnya sesuai norma yang berlaku di perguruan tinggi.

Pekanbaru, 26 Mei 2024  
Yang membuat pernyataan

Sammi Aldhi Yanto  
NIM. 2003113948

## KATA PENGANTAR

Alhamdulillahirobbil'alamin segala puji dan syukur penulis sampaikan atas kehadirat Allah Subhanahu wa Ta'ala, yang telah memberikan pertolongan, rahmat dan hidayah-Nya. Shalawat serta salam senantiasa tercurahkan kepada rasul pemimpin umat yang telah membawa agama islam sebagai petunjuk bagi umat manusia, yakni baginda Nabi Muhammad Shalaallahu 'alaihi wassalam. Berkat ridha Allah Subhanahu wa Ta'ala, penulis mampu menyusun dan menyelesaikan skripsi yang berjudul "**Pengembangan Auto Grader Menggunakan Metode Domain Driven Design di Jurusan Ilmu Komputer Universitas Riau**", sebagai salah satu persyaratan untuk memperoleh gelar sarjana di Jurusan Ilmu Komputer Universitas Riau.

Selama proses penelitian dan penyusunan skripsi penulis telah mendapatkan pengetahuan, bimbingan, dukungan dan saran dari berbagai pihak yang telah bersedia membantu dari awal hingga akhir. Oleh sebab itu, penulis ingin mengucapkan terima kasih kepada Bapak Dr. Syamsudhuha, M.Sc. selaku Dekan FMIPA Universitas Riau, kepada Bapak Roni Salambue, S.Kom., M.Si. selaku Ketua Jurusan Ilmu Komputer FMIPA Universitas Riau, kepada Bapak Drs. Sukamto, M.Kom. selaku Koordinator Program Studi Sistem Informasi FMIPA Universitas Riau sekaligus Dosen Pembimbing Akademik, kepada Ibu Yanti Andriyani, S.T., M.T.I., Ph.D. selaku Dosen Pembimbing yang telah memberikan arahan, nasihat, dan dukungan kepada penulis sehingga penulisan skripsi ini dapat diselesaikan.

Penulis juga mengucapkan banyak terima kasih kepada Ibu Tirta Mailinda, S.T. selaku Staf Tata Usaha Program Studi Sistem Informasi FMIPA Universitas Riau. Teristimewa penulis mengucapkan terima kasih yang sebesar-besarnya kepada ayah, ibu serta keluarga yang selalu memberikan dukungan, nasehat, dan doa sehingga penulis dapat berada pada posisi saat sekarang ini.

Akhir kata, semoga skripsi ini dapat memberikan kebermanfaatan bagi pembaca dan khalayak ramai serta menjadi kontribusi dalam pengembangan ilmu

pengetahuan terkait. Namun di luar itu, penulis menyadari bahwa skripsi ini tidak luput dari kekurangan dan kekhilafan baik dalam segi materi maupun penyajiannya, sehingga segala saran dan kritik yang bersifat membangun dari pembaca akan sangat membantu penulis dalam berkarya untuk kedepannya. Demikian kata pengantar ini ditulis, penulis mengucapkan terimakasih dan selamat membaca.

*Wassalamu'alaikum Warahmatullaahi Wabarakaaatu*

Pekanbaru, Maret 2024

Sammi Aldhi Yanto

NIM. 2003113948

## ABSTRAK

Mata kuliah praktikum pemrograman seperti Konsep Pemrograman dan Pemrograman Berorientasi Objek di Jurusan Ilmu Komputer Universitas Riau sering kali memerlukan penulisan kode program dalam tugas-tugasnya. Dengan jumlah mahasiswa yang cukup banyak mengambil mata kuliah tersebut, proses penilaian manual menjadi rumit dan memakan waktu bagi pengajar. Untuk mengatasi hal ini, dikembangkan aplikasi *auto grader* berbasis web yang berfungsi untuk menilai tugas pemrograman secara otomatis, memungkinkan penilaian program-program secara efisien dalam jumlah yang banyak. Pengembangan aplikasi *auto grader* menggunakan metode *domain driven design* yang berfokus pada solusi perangkat lunak dalam bentuk model yang mencerminkan inti dari masalah yang sedang diselesaikan. Metode ini menyediakan sejumlah prinsip dan teknik sistematis untuk menganalisis, merancang, dan mengimplementasikan solusi perangkat lunak. Tahap pertama dalam metode ini adalah menentukan *domain*, menerapkan dua pilar *domain driven design* yaitu *strategic design* dan *ubiquitous language* yang menghasilkan *sub-domain*, *ubiquitous language*, *bounded context*, *domain model*, dan *context map*. Tahap berikutnya adalah menentukan dan menerapkan *tactical design* yang mencakup *entities*, *aggregates*, *value objects*, *factories*, *repositories*, dan *services*. Hasil dari penelitian ini menunjukkan bahwa *user acceptance test* mendapatkan skor 80%, yang berdasarkan hasil total persentase aplikasi dan kriteria interpretasi skor masuk dalam kategori baik.

**Kata Kunci:** *Auto Grader, Domain Driven Design, Strategic Design, Ubiquitous Language, Tactical Design.*

## ***ABSTRACT***

*Programming practicum courses such as Programming Concepts and Object-Oriented Programming in the Computer Science Department at the University of Riau often require coding assignments. With a significant number of students enrolled in these courses, manual grading becomes complex and time-consuming for instructors. To address this issue, a web-based auto-grader application was developed to automatically evaluate programming assignments, allowing for efficient grading of a large number of programs. The development of the auto-grader application employed the domain-driven design (DDD) method, which focuses on software solutions in the form of models that reflect the core of the problem being solved. This method provides a set of systematic principles and techniques for analyzing, designing, and implementing software solutions. The first stage of this method involves defining the domain, applying the two pillars of DDD, namely strategic design and ubiquitous language, which result in sub-domains, ubiquitous language, bounded context, domain model, and context map. The next stage involves defining and applying tactical design, which includes entities, aggregates, value objects, factories, repositories, and services. The results of this research indicated that the user acceptance test scored 80%, placing the application in the "good" category according to the score interpretation criteria.*

***Keywords:*** *Auto Grader, Domain Driven Design, Strategic Design, Ubiquitous Language, Tactical Design.*

## **DAFTAR ISI**

	Halaman
<b>HALAMAN JUDUL .....</b>	<b>i</b>
<b>HALAMAN PERSETUJUAN PEMBIMBING .....</b>	<b>ii</b>
<b>HALAMAN PERSETUJUAN TIM PENGUJI .....</b>	<b>iii</b>
<b>LEMBAR PERNYATAAN .....</b>	<b>iv</b>
<b>KATA PENGANTAR.....</b>	<b>v</b>
<b>ABSTRAK .....</b>	<b>vii</b>
<b>ABSTRACT .....</b>	<b>viii</b>
<b>DAFTAR ISI.....</b>	<b>ix</b>
<b>DAFTAR GAMBAR.....</b>	<b>xii</b>
<b>DAFTAR TABEL.....</b>	<b>xiv</b>
<b>BAB I PENDAHULUAN .....</b>	<b>1</b>
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	2
1.3 Batasan Masalah.....	2
1.4 Tujuan dan Manfaat Penelitian .....	3
<b>BAB II TINJAUAN PUSTAKA.....</b>	<b>4</b>
2.1 Pengembangan Aplikasi .....	4
2.2 Aplikasi .....	4
2.3 <i>Auto Grader</i> .....	5
2.4 <i>Domain Driven Design</i> .....	7

2.4.1 <i>Strategic Design</i> .....	7
2.4.2 <i>Domain and Technical Experts</i> .....	9
2.4.3 <i>Ubiquitous Language</i> .....	9
2.4.4 <i>Tactical Design</i> .....	12
2.5 <i>Dynamic Approach</i> .....	14
2.6 <i>Go</i> .....	15
2.7 <i>Laravel</i> .....	15
2.8 <i>MongoDB</i> .....	16
2.9 <i>Unified Modeling Language (UML)</i> .....	16
2.9.1 <i>Use Case Diagram</i> .....	17
2.9.2 <i>Activity Diagram</i> .....	18
2.9.3 <i>Sequence Diagram</i> .....	19
2.9.4 <i>Class Diagram</i> .....	20
2.10 <i>Black Box Testing</i> .....	21
2.11 <i>User Acceptance Testing</i> .....	22
2.12 Penelitian Terdahulu .....	24
<b>BAB III METODE PENELITIAN .....</b>	<b>26</b>
3.1 Teknik Pengumpulan Data .....	26
3.2 Peralatan yang Digunakan.....	27
3.3 Langkah-Langkah Penyelesaian.....	27
<b>BAB IV HASIL DAN PEMBAHASAN.....</b>	<b>31</b>
4.1 Pengumpulan Data dan Analisis Kebutuhan .....	31
4.2 Perancangan dan <i>Design</i> Perangkat Lunak .....	33
4.2.1 <i>Domain dan Sub-Domain</i> .....	33
4.2.2 <i>Bounded Context</i> .....	34
4.2.3 Daftar Istilah .....	35
4.2.4 <i>Context Map</i> .....	36

4.2.5 <i>Domain Model</i> .....	36
4.2.6 <i>Use Case Diagram</i> .....	37
4.2.7 <i>Activity Diagram</i> .....	48
4.2.8 <i>Sequence Diagram</i> .....	54
4.2.9 <i>Class Diagram</i> .....	57
4.2.10 <i>User Interface Design</i> .....	58
4.3 Implementasi .....	64
4.3.1 <i>Entities</i> .....	64
4.3.2 <i>Value Objects</i> .....	66
4.3.3 <i>Aggregates</i> .....	66
4.3.4 <i>Factories</i> .....	67
4.3.5 <i>Repositories</i> .....	68
4.3.6 <i>Services</i> .....	69
4.3.7 <i>User Interface</i> .....	70
4.4 Pengujian.....	76
4.4.1 <i>Black Box Testing</i> .....	76
4.4.2 <i>User Acceptance Testing</i> .....	80
4.5 Deployment.....	83
4.6 Maintenance .....	84
<b>BAB V KESIMPULAN DAN SARAN .....</b>	<b>86</b>
5.1 Kesimpulan.....	86
5.2 Saran.....	87
<b>DAFTAR PUSTAKA .....</b>	<b>89</b>
<b>LAMPIRAN.....</b>	<b>92</b>

## DAFTAR GAMBAR

	Halaman
<b>Gambar 2.1 Tampilan Aplikasi <i>LeetCode</i> .....</b>	<b>6</b>
<b>Gambar 3.1 Bagan Tahapan Penelitian .....</b>	<b>28</b>
<b>Gambar 4.1 Sub Domain <i>Auto Grader</i> .....</b>	<b>34</b>
<b>Gambar 4.2 Bounded Context <i>Auto Grader</i> .....</b>	<b>34</b>
<b>Gambar 4.3 Context Map.....</b>	<b>36</b>
<b>Gambar 4.4 Domain Model .....</b>	<b>37</b>
<b>Gambar 4.5 Use Case Diagram.....</b>	<b>38</b>
<b>Gambar 4.6 Activity Diagram Autentikasi .....</b>	<b>49</b>
<b>Gambar 4.7 Activity Diagram Membuat Kelas .....</b>	<b>50</b>
<b>Gambar 4.8 Activity Diagram Bergabung Menggunakan Kode Kelas .....</b>	<b>52</b>
<b>Gambar 4.9 Activity Diagram Membuat Tugas.....</b>	<b>52</b>
<b>Gambar 4.10 Activity Diagram Membuat Submission .....</b>	<b>53</b>
<b>Gambar 4.11 Sequence Diagram Autentikasi .....</b>	<b>54</b>
<b>Gambar 4.12 Sequence Diagram Membuat Kelas .....</b>	<b>55</b>
<b>Gambar 4.13 Sequence Diagram Bergabung Menggunakan Kode Kelas.....</b>	<b>55</b>
<b>Gambar 4.14 Sequence Diagram Membuat Tugas.....</b>	<b>56</b>
<b>Gambar 4.15 Sequence Diagram Membuat Submission.....</b>	<b>57</b>
<b>Gambar 4.16 Class Diagram .....</b>	<b>57</b>
<b>Gambar 4.17 Wireframe Halaman Autentikasi .....</b>	<b>58</b>
<b>Gambar 4.18 Wireframe Halaman Kelas.....</b>	<b>59</b>
<b>Gambar 4.19 Wireframe Halaman Tugas .....</b>	<b>60</b>

<b>Gambar 4.20</b> <i>Wireframe</i> Halaman Buat Tugas .....	60
<b>Gambar 4.21</b> <i>Wireframe</i> Halaman Anggota Kelas.....	61
<b>Gambar 4.22</b> <i>Wireframe</i> Halaman Pengaturan Kelas.....	62
<b>Gambar 4.23</b> <i>Wireframe</i> Halaman Daftar <i>Submission</i> Siswa.....	62
<b>Gambar 4.24</b> <i>Wireframe</i> Halaman Daftar Riwayat Pengumpulan Tugas .....	63
<b>Gambar 4.25</b> <i>Wireframe</i> Halaman Papan Peringkat.....	64
<b>Gambar 4.26</b> Halaman Autentikasi .....	70
<b>Gambar 4.27</b> Halaman Kelas.....	71
<b>Gambar 4.28</b> Halaman Daftar Tugas.....	71
<b>Gambar 4.29</b> Halaman Buat Tugas.....	72
<b>Gambar 4.30</b> Halaman Daftar Anggota Kelas.....	73
<b>Gambar 4.31</b> Halaman Pengaturan Kelas.....	73
<b>Gambar 4.32</b> Halaman <i>Submission</i> Mahasiswa .....	74
<b>Gambar 4.33</b> Halaman Riwayat <i>Submission</i> .....	75
<b>Gambar 4.34</b> Halaman Peringkat.....	75
<b>Gambar 4.35</b> Spesifikasi <i>Server</i> .....	84
<b>Gambar 4.36</b> <i>Resource Monitoring</i> .....	85

## DAFTAR TABEL

	Halaman
<b>Tabel 2.1</b> Simbol-simbol <i>Use Case Diagram</i> .....	17
<b>Tabel 2.2</b> Simbol-simbol <i>Activity Diagram</i> .....	19
<b>Tabel 2.3</b> Simbol-simbol <i>Sequence Diagram</i> .....	20
<b>Tabel 2.4</b> Simbol-simbol <i>Class Diagram</i> .....	21
<b>Tabel 2.5</b> Skala <i>Likert</i> .....	23
<b>Tabel 2.6</b> Kriteria Interpretasi Skor.....	23
<b>Tabel 4.1</b> Daftar <i>Functional Requirements Auto Grader</i> .....	32
<b>Tabel 4.2</b> Daftar Istilah atau <i>Ubiquitous Language</i> .....	35
<b>Tabel 4.3</b> Deskripsi <i>Use Case Login</i> .....	39
<b>Tabel 4.4</b> Deskripsi <i>Use Case Membuat Kelas</i> .....	40
<b>Tabel 4.5</b> Deskripsi <i>Use Case Bergabung Menggunakan Kode Kelas</i> .....	41
<b>Tabel 4.6</b> Deskripsi <i>Use Case Melihat Detail Tugas</i> .....	42
<b>Tabel 4.7</b> Deskripsi <i>Use Case Melihat Leaderboard</i> .....	43
<b>Tabel 4.8</b> Deskripsi <i>Use Case Melihat Riwayat Submission</i> .....	44
<b>Tabel 4.9</b> Deskripsi <i>Use Case Melihat Anggota Kelas</i> .....	45
<b>Tabel 4.10</b> Deskripsi <i>Use Case Membuat Submission</i> .....	46
<b>Tabel 4.11</b> Deskripsi <i>Use Case Membuat Tugas</i> .....	47
<b>Tabel 4.12</b> Hasil <i>Testing</i> Fitur Autentikasi.....	76
<b>Tabel 4.13</b> Hasil <i>Testing</i> Fitur Kelas.....	77
<b>Tabel 4.14</b> Hasil <i>Testing</i> Fitur Anggota Kelas .....	78

<b>Tabel 4.15</b> Hasil <i>Testing Fitur Tugas</i> .....	79
<b>Tabel 4.16</b> Pertanyaan Kuisioner <i>User Acceptance Testing</i> .....	82
<b>Tabel 4.17</b> Hasil UAT Responden.....	82
<b>Tabel 4.18</b> Hasil Presentase Pertanyaan UAT .....	83

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang**

Jurusan Ilmu Komputer Universitas Riau memiliki Prodi Sistem Informasi dan Manajemen Informatika yang berfokus pada pengembangan dan penerapan teknologi informasi di berbagai bidang. Mata kuliah praktikum pemrograman seperti Konsep Pemrograman dan Pemrograman Berorientasi Objek seringkali memberikan tugas-tugas pemrograman yang melibatkan penulisan kode program. Dengan jumlah mahasiswa yang cukup banyak yang mengambil mata kuliah tersebut, proses penilaian manual menjadi rumit dan memakan waktu bagi pengajar (Combéfis, 2022). Oleh karena itu, *auto grader* menjadi solusi yang diperlukan untuk mengatasi permasalahan ini.

*Auto grader* adalah aplikasi dapat melakukan proses penilaian secara otomatis, menjadikan penilaian lebih akurat dan efisien karena tidak memerlukan campur tangan manual dari pengajar (Caiza & Alamo, 2013). Selain itu, *auto grader* dapat memberikan hasil penilaian yang adil dan objektif dengan menggunakan kriteria yang telah ditetapkan sebelumnya, sehingga mengurangi bias subjektivitas dalam penilaian.

Dalam pengembangan aplikasi *auto grader*, penerapan metode *domain driven design* menjadi relevan karena memberikan kerangka kerja yang memungkinkan pemodelan yang kuat, meliputi pembentukan konsep-konsep seperti *aggregates*, *entities*, *value objects*, dan *services* (Khononov & Lerman,

2022) dan berfokus pada kolaborasi antara *technical experts* dan *domain experts*. Selain itu, metode ini juga memungkinkan pemahaman mendalam tentang *domain* aplikasi, yang membantu pengembang dalam menangkap dengan tepat kebutuhan dan kompleksitas dari domain tersebut.

Dengan demikian, penggunaan *auto grader* diharapkan dapat memberikan kemudahan dan efisiensi dalam proses penilaian tugas-tugas pemrograman di lingkungan akademis, khususnya di Jurusan Ilmu Komputer Universitas Riau. Melalui penerapan metode *domain driven design*, pengembangan *auto grader* dapat dilakukan dengan pemahaman yang mendalam tentang kebutuhan dan kompleksitas dari *domain* aplikasi, sehingga menghasilkan aplikasi yang sesuai dan bermanfaat.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka masalah yang akan dibahas pada penelitian adalah sebagai berikut:

1. Bagaimana membangun aplikasi *auto grader* agar dapat melakukan penilaian tugas pemrograman secara otomatis, elemen apa saja yang dibutuhkan dalam perancangan dan pengembangan aplikasi *auto grader*?
2. Bagaimana penerapan metode *domain driven design* dalam pengembangan aplikasi *auto grader*?

## 1.3 Batasan Masalah

Penelitian yang akan dilakukan memiliki batasan yaitu:

1. Aplikasi dikembangkan sebagai *platform web* yang fokus pada penilaian bahasa pemrograman *Java*.
2. Metode *domain driven design* digunakan dalam pengembangan aplikasi.
3. Penilaian otomatis menggunakan pendekatan dinamis atau *test-based assessment*.
4. Pengembangan aplikasi dilakukan menggunakan bahasa pemrograman *Go* dan *framework Laravel*.

#### **1.4 Tujuan dan Manfaat Penelitian**

Adapun tujuan dari penelitian adalah sebagai berikut:

1. Untuk membangun aplikasi *auto grader* yang dapat membantu dosen dalam melakukan penilaian tugas pemrograman mahasiswa secara otomatis dan efektif.
2. Untuk memahami penerapan metode *domain driven design* dalam mengembangkan aplikasi *auto grader*.

Adapun manfaat dari penelitian ini adalah:

1. Bagi peneliti: Meningkatkan pemahaman dalam membangun *auto grader* menggunakan metode *domain driven design* dan menambah portofolio dengan hasil kontribusi yang dapat diakses dan diaplikasikan.
2. Bagi prodi: Berguna sebagai dasar acuan, serta sumber tambahan literatur baru untuk penelitian yang lebih lanjut atau serupa.
3. Bagi khalayak ramai: Membantu pengguna dalam proses penilaian tugas pemrograman dengan lebih objektif dan efisien.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Pengembangan Aplikasi**

Pengembangan aplikasi merupakan kegiatan yang melibatkan lebih dari sekedar menulis kode, melainkan serangkaian proses yang meliputi perencanaan awal, pembuatan *prototype*, implementasi, pengujian, dan *maintenance*. Tahapan-tahapan tersebut menjadi kunci dalam memastikan aplikasi dibangun secara terstruktur dan terencana. Metode-metode pengembangan aplikasi seperti *Waterfall*, *Spiral*, *XP*, *Kanban*, *Agile*, *Prototype*, dan *Scrum* memungkinkan *programmer* untuk memilih pendekatan yang sesuai dengan kebutuhan aplikasi (Rahmi dkk., 2014). Dalam penelitian ini, pendekatan yang akan diterapkan adalah kombinasi metode *waterfall* dengan *domain driven design*.

#### **2.2 Aplikasi**

Aplikasi merupakan satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas seperti *game*, sistem perniagaan, periklanan, pelayanan masyarakat, dan hamper semua proses kegiatan (Nurjanah & Zulfikli, 2017). Aplikasi bisa berupa program yang berdiri sendiri atau kumpulan program yang terdiri dari beberapa fitur khusus yang dapat diakses oleh pengguna. Jenis-jenis aplikasi berdasarkan platformnya adalah sebagai berikut:

### 1. Aplikasi *Desktop*

Aplikasi *desktop* adalah program yang dikembangkan untuk dijalankan di lingkungan sistem operasi *desktop* seperti *Windows*, *macOS*, atau *Linux*. Aplikasi ini umumnya dilengkapi dengan antarmuka pengguna yang disesuaikan khusus untuk digunakan pada komputer atau laptop.

### 2. Aplikasi *Website*

Aplikasi *website* adalah program yang dapat diakses melalui peramban internet tanpa perlu instalasi langsung pada perangkat pengguna. Aplikasi ini dapat diakses dari berbagai perangkat dengan koneksi internet dan kompatibel dengan berbagai sistem operasi seperti *Windows*, *macOS*, *Linux*, *Android*, dan *iOS*.

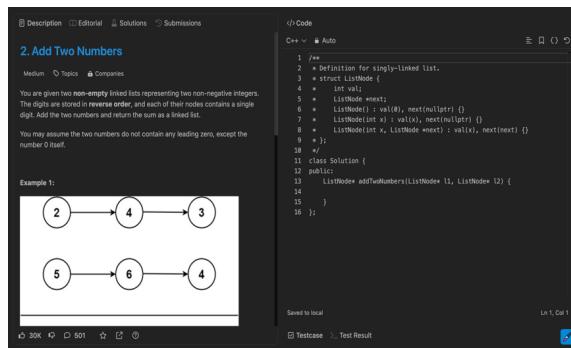
### 3. Aplikasi *Mobile*

Aplikasi *mobile* adalah program yang didesain khusus untuk perangkat seluler seperti *smartphone* dan *tablet*. Aplikasi ini dapat diunduh dan diinstal langsung pada perangkat melalui toko aplikasi resmi seperti *Google Play Store* untuk *Android* atau *App Store* untuk *iOS*. Aplikasi *mobile* beroperasi di sistem operasi tertentu, seperti *Android*, *iOS*, atau *Windows Phone*.

## 2.3 Auto Grader

*Auto grader* atau penilaian secara otomatis adalah aplikasi yang dikembangkan untuk menilai suatu pekerjaan secara otomatis yang dalam konteks ini kode program. Aplikasi *auto grader* memungkinkan penilaian program-program secara efisien dalam jumlah yang banyak.

Contoh aplikasi *auto grader* yang sudah ada seperti *HackerRank*, *LeetCode*, dan *CodeSignal* (Michail & Pervolarakis, 2023). Tampilan aplikasi *leetcode* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Tampilan Aplikasi *LeetCode*

Keunggulan aplikasi *auto grader* seperti *HackerRank* dan *LeetCode* adalah kemampuannya untuk memberikan tantangan pemrograman dengan berbagai tingkat kesulitan serta menyediakan uji coba langsung secara *online*. Untuk aplikasi yang akan dibuat, pembeda utamanya adalah fokus pada lingkungan Pendidikan dengan menyediakan fitur-fitur yang mendukung pengajaran dan pembelajaran secara efektif mirip aplikasi *LMS* seperti *Classroom*. Beberapa fitur yang menarik yang ditambahkan seperti fitur riwayat *submissions*, manajemen kelas agar pengajar dapat mengatur dan mengorganisir kelas dengan lebih efisien, manajemen tugas, mengundang pengguna lain ke dalam kelas dengan menggunakan kode kelas, rekapitulasi nilai, *leaderboard* atau peringkat dari suatu tugas sehingga pengguna dapat memantau kemajuan dan memotivasi siswa dalam pembelajaran.

## **2.4 Domain Driven Design**

*Domain Driven Design* pertama kali diperkenalkan oleh Eric Evans dalam bukunya tahun yang berjudul “*Domain-Driven Design: Tackling Complexity in the Heart of Software*” yang dirilis pada tahun 2003 (Khononov & Lerman, 2022). *Domain driven design* adalah pendekatan dalam pengembangan perangkat lunak yang berfokus pada solusi perangkat lunak dalam bentuk model yang mencerminkan inti dari masalah yang sedang diselesaikan. *Domain driven design* berfokus pada kolaborasi antara *technical experts* dan *domain experts* (Prayoga & Akbar, 2021). *Domain driven design* menyediakan sejumlah prinsip dan Teknik sistematis untuk menganalisis, merancang, dan mengimplementasikan solusi perangkat lunak.

Terdapat tiga konsep utama atau pilar *domain driven design* yaitu *Strategic Design*, *Ubiquitous language*, dan *Tactical Design* (Boyle Matthew, 2022).

### **2.4.1 Strategic Design**

*Strategic design* adalah fase dalam proses *Domain driven design* di mana dilakukan pemetaan *business domain* dan mendefinisikan *bounded context*. Tujuan dari *strategic design* adalah memastikan bahwa arsitektur sistem yang dirancang berfokus pada *business outcome* atau hasil (Boyle Matthew, 2022). Hal ini dicapai dengan membuat *domain model*. *Domain model* merupakan representasi struktural dari suatu *domain bisnis*. *Domain model* dibangun berdasarkan pemahaman yang mendalam terhadap *domain bisnis* yang sedang dihadapi.

#### **2.4.1.1 *Domain***

*Domain* merujuk pada cakupan keseluruhan dari aktivitas utama suatu bisnis atau layanan yang diberikan kepada pelanggan. Ini juga dikenal sebagai *problem domain* atau ranah masalah (Chandrasekaran & Krishnan, 2022). Contohnya, *Tesla* beroperasi dalam *domain* kendaraan listrik, *Netflix* menyediakan film dan acara *online*, dan *McDonald's* menyediakan makanan cepat saji. Beberapa perusahaan, seperti *Amazon*, menyediakan layanan dalam lebih dari satu *domain*, yaitu penjualan *online* dan komputasi awan. Untuk mengatasi kompleksitas bisnis *domain*, *domain-domain* ini dibagi menjadi bagian-bagian yang lebih mudah dikelola yang disebut *subdomain*

#### **2.4.1.2 *Sub Domain***

*Subdomain* merujuk pada cara untuk mengatasi kompleksitas dengan memecah masalah yang kompleks menjadi masalah yang lebih mudah dikelola (Chandrasekaran & Krishnan, 2022). *Subdomain* membantu dalam memudahkan pemahaman dan mempermudah pencarian solusi. Sebagai contoh, dalam *domain* penjualan *online*, *subdomain* dapat dibagi menjadi area seperti *products*, *inventory*, *shopping carts*, *payments*, *activity tracking*, *order management*, dan *shipping*. Setiap *subdomain* ini mewakili fokus khusus dalam *domain* yang lebih besar, dan pendekatan ini membantu dalam pengembangan yang lebih terfokus dan efisien.

#### **2.4.2 Domain and Technical Experts**

*Domain expert* adalah individu yang memiliki pemahaman yang dalam dan mendalam tentang suatu domain bisnis tertentu. Mereka adalah pakar dalam bidang tersebut dan sering disebut sebagai *Subject-Matter Experts* (SMEs) yang memiliki pemahaman yang kuat tentang aspek bisnis (Chandrasekaran & Krishnan, 2022). *Domain expert* bertanggung jawab untuk “mengapa” dan “apa” yang perlu dicapai. Sedangkan, *Technical expert* adalah individu yang ahli dalam menyelesaikan masalah-masalah dalam ilmu komputer yang bersifat teknis. *Technical expert* memiliki peran besar dalam mewujudkan “bagaimana” suatu solusi teknis dapat diimplementasikan.

#### **2.4.3 Ubiquitous Language**

*Ubiquitous language* adalah istilah yang digunakan untuk menggambarkan proses membangun bahasa bersama yang dapat digunakan ketika berbicara tentang *domain* (Boyle Matthew, 2022). Bahasa ini digunakan oleh semua anggota tim, termasuk pengembang dan individu dari pihak bisnis. *Ubiquitous language* ini mengumpulkan tim dengan memastikan bahwa tidak ada ketidakjelasan dalam komunikasi. Dengan kata lain, *ubiquitous language* adalah cara untuk memastikan bahwa seluruh tim memiliki pemahaman yang seragam tentang istilah dan konsep yang digunakan dalam *domain* bisnis yang sedang dikerjakan. Ini membantu menghindari salah paham dan memastikan bahwa semua pihak dapat berkomunikasi dengan jelas dan efektif ketika berdiskusi tentang proyek atau solusi perangkat lunak yang sedang dikembangkan.

#### **2.4.3.1 Domain Model**

*Domain model* adalah representasi dari solusi dalam perangkat lunak yang dibuat berdasarkan bahasa bersama (*ubiquitous language*) yang telah disepakati oleh anggota tim. Model *domain* ini digunakan untuk meningkatkan pemahaman yang konsisten di antara anggota tim tentang bagaimana masalah bisnis tertentu akan diimplementasikan dalam bentuk perangkat lunak (Khononov & Lerman, 2022).

#### **2.4.3.2 Bounded Context dan Context Map**

*Bounded context* adalah batasan yang digunakan untuk membatasi cakupan dari *domain model* yang dibuat dalam pengembangan perangkat lunak. Saat menggunakan *domain driven design*, sistem secara keseluruhan dapat direpresentasikan sebagai himpunan yang dibatasi konteks yang mempunyai hubungan satu sama lain.

*Context map* adalah representasi visual dari hubungan antara berbagai *bounded context*. Meskipun *bounded contexts* dirancang untuk sebebas mungkin, namun mungkin masih diperlukan komunikasi antara *bounded context* tersebut. Dalam DDD, sistem secara keseluruhan dapat diwakili sebagai kumpulan *bounded contexts* yang memiliki hubungan satu sama lain. Hubungan ini mendefinisikan cara-cara di mana *bounded contexts* tersebut dapat terintegrasi satu sama lain. Menurut (Boyle Matthew, 2022) terdapat 4 pattern utama dalam untuk mengintegrasikan antar bounded context:

1. *Shared Kernel*

*Shared Kernel* merupakan situasi di mana dua atau lebih *bounded contexts* sepakat untuk menggunakan model atau artefak tertentu secara bersamaan (Khononov & Lerman, 2022). Dalam hal ini, tim-tim tersebut memiliki pemahaman yang jelas tentang model atau artefak solusi yang mereka pilih untuk saling berbagi, dan keduanya memiliki tanggung jawab yang sama terhadap pemeliharaan artefak-artefak tersebut.

2. *Open Host Service*

*Open Host Service* merujuk pada mekanisme yang memungkinkan sistem atau *sub-sistem* lain untuk mengakses sistem. *Open Host Service* umumnya diimplementasikan sebagai *RPC*, dan beberapa opsi yang dapat dipertimbangkan untuk *RPC* mencakup pembangunan *RESTful API*, penerapan *gRPC*, atau bahkan penggunaan *XML API* (Boyle Matthew, 2022). Dengan menerapkan *Open Host Service*, akses terbuka diberikan kepada sistem atau layanan kepada entitas lain, memungkinkan interaksi antara komponen-komponen dengan pendekatan yang sesuai dengan kebutuhan dan kendala yang ada.

3. *Published Language*

*Published Language* merujuk pada bahasa yang secara resmi didefinisikan untuk mengkomunikasikan informasi atau layanan yang diekspos oleh suatu tim kepada tim lain melalui *Open Host Service* (Boyle Matthew, 2022). Berbeda dengan *ubiquitous language* yang merupakan bahasa *internal* tim, *published language* memiliki tujuan untuk memastikan bahwa definisi dari

apa yang diekspos kepada tim lain dalam berbagai konteks terbatas jelas dan dapat dipahami. Sebagai contoh, jika sebuah tim membuat *server HTTP* untuk memiliki *endpoint GET /{id}/user*, maka tim perlu membuat *published language* untuk membantu tim lain memahami skema keluaran. Dua cara populer untuk menyajikan *published language* adalah melalui *OpenAPI* atau *gRPC*.

#### 4. *Anti-Corruption Layer*

*Anti-corruption layer*, juga dikenal sebagai *adapter layer*, merupakan suatu lapisan yang digunakan untuk menerjemahkan model dari sistem yang berbeda. Ini adalah pola yang melengkapi dan bekerja secara baik dengan *Open Host Service* (Khononov & Lerman, 2022).

### **2.4.4 *Tactical Design***

Desain taktis adalah fase dalam *Domain Driven Design* di mana perhatian diberikan pada rincian khusus tentang implementasi sistem. *Tactical design* mencakup *Entities*, *Aggregates*, *Value Objects*, *Factories*, *Services*, dan *Repositories* (Boyle Matthew, 2022).

#### **2.4.4.1 *Entities***

*Entities* adalah objek yang didefinisikan oleh identitasnya. Identitas adalah atribut yang unik dan khas yang membedakan entitas tersebut dari entitas lain (Chandrasekaran & Krishnan, 2022). Meskipun atribut-atribut dari sebuah entitas dapat berubah seiring waktu, identitasnya diharapkan untuk tetap konstan.

#### **2.4.4.2 *Value Objects***

*Value object* adalah elemen dalam *Domain Driven Design* yang memiliki sifat yang berlawanan dengan entitas dalam beberapa aspek. Dalam konteks objek nilai, tujuan utama adalah untuk memastikan bahwa dua objek dianggap identik jika nilai-nilai *internal* mereka sama (Boyle Matthew, 2022). Konsep objek nilai ini tidak mengakui adanya identitas unik yang membedakan entitas, sebaliknya, objek nilai diidentifikasi dan dibandingkan berdasarkan nilai-nilai yang mereka bawa. *Value object* bersifat *immutable*, artinya *state* dari *attribute* tidak dapat diubah setelah *value object* itu dibuat pertama kali.

#### **2.4.4.3 *Aggregates***

*Aggregat* mengacu pada sekelompok objek *domain* yang dapat ditangani sebagai satu kesatuan (Boyle Matthew, 2022). Aggregat terdiri dari *entities* dan *value objects* yang membentuk hierarki komposisi dalam strukturnya. Aggregat juga berfungsi sebagai batas yang memegang peran krusial dalam menjaga konsistensi dan menerapkan aturan bisnis secara konsisten dalam *bounded context*.

#### **2.4.4.4 *Factories***

*Factories* merupakan mekanisme yang digunakan untuk mengkonstruksi instansiasi dari *Aggregates* dan *Value Objects* (Khononov & Lerman, 2022). Dalam kasus yang sederhana, konstruksi dapat dilakukan menggunakan konstruktor biasa.

Namun, instansiasi agregat dan objek nilai dapat menjadi kompleks tergantung pada jumlah data.

#### **2.4.4.5 *Repositories***

*Repositories* adalah bagian dari kode yang mengandung logika yang diperlukan untuk mengakses berbagai sumber data, seperti file di *disk*, *spreadsheet*, atau penyimpanan *AWS S3* (Khononov & Lerman, 2022), walaupun pada umumnya berupa *database*. Dengan menggunakan lapisan *repository* akan membuat sistem lebih mudah dipelihara dengan memisahkan dari teknologi *database* tertentu.

#### **2.4.4.6 *Services***

*Service* merujuk pada objek atau komponen yang digunakan untuk menyediakan fungsionalitas tertentu yang lebih kompleks dan memerlukan interaksi dengan beberapa *bounded context* dan *aggregate* (Khononov & Lerman, 2022).

### **2.5 *Dynamic Approach***

*Dynamic Approach* adalah metode yang melibatkan eksekusi kode program sebagai bagian dari proses evaluasi. Pendekatan ini umumnya berdasarkan analisis hasil keluaran yang dihasilkan oleh eksekusi kode dan membandingkannya dengan solusi referensi yang diharapkan. Biasanya, Library seperti *JUnit* digunakan untuk menerapkan pendekatan ini dalam program *Java* (Aldriye dkk., 2019)

## **2.6 Go**

*Go* atau yang lebih sering dikenal dengan *Golang* adalah bahasa pemrograman yang diciptakan dan dikembangkan oleh tim *Engineer Google* pada tahun 2009. Awalnya bahasa tersebut hanya digunakan untuk kepentingan *internal*. Kemudian Bahasa ini dirilis untuk kepentingan *public* dan bersifat *Open Source* sehingga siapapun bisa mengembangkan bahasa *Go*. Bahasa pemrograman *Go* memiliki beberapa kegunaan diantaranya sebagai bahasa dalam membangun *backend Stack*, pengembangan aplikasi *E-Commerce*, dan pengembangan *cloud native*. *Go* memiliki beberapa kelebihan yaitu mempunyai fitur *concurrency*, proses kompilasi yg cepat, dan mempunyai *garbage collector* (Putri, 2023). Dalam *auto grader* yang akan dibangun, bahasa pemrograman *Go* berfokus pada bagian *backend stack*.

## **2.7 Laravel**

*Laravel* merupakan sebuah kerangka kerja web berbasis *PHP* yang bersifat *open-source* dan gratis, dikembangkan oleh Taylor Otwell untuk memfasilitasi pengembangan aplikasi *web* yang mengikuti pola *MVC* (Model-View-Controller) (Endra dkk., 2021). Struktur pola *MVC* pada *Laravel* sedikit berbeda dari struktur pola *MVC* pada umumnya. Dalam *Laravel*, terdapat sistem *routing* yang bertindak sebagai penghubung antara *request* dari pengguna dan kontroler.

## **2.8 MongoDB**

*MongoDB* adalah sistem basis data *NoSQL* (*Not Only SQL*) yang dikembangkan untuk menyimpan dan mengelola data dalam format dokumen yang fleksibel. Disebut juga sebagai basis data dokumen, *MongoDB* menyimpan data dalam format  *BSON (Binary JSON)*, yang merupakan representasi *biner* dari *JSON (JavaScript Object Notation)*.

Salah satu ciri khas *MongoDB* adalah kemampuannya untuk menyimpan dan mengelola data semi-struktural atau tidak terstruktur, sehingga memungkinkan pengembang untuk menyimpan data dengan skema yang berbeda di dalam koleksi dokumen.

## **2.9 Unified Modeling Language (UML)**

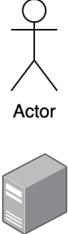
*Unified Modeling Language (UML)* adalah sebuah bahasa pemodelan visual umum yang sering digunakan untuk mengidentifikasi, memvisualisasikan, membangun, dan mendokumentasikan elemen-elemen dari sistem perangkat lunak (Adhiwibowo & Daru, 2017). Dengan *UML*, pengembang dapat menangkap keputusan dan pemahaman tentang sistem yang sedang dibangun. *UML* juga menyediakan kumpulan simbol grafis yang dapat digunakan untuk menggambarkan berbagai aspek dalam perancangan perangkat lunak, termasuk kelas, objek, hubungan antar-objek, proses bisnis, penggunaan kasus, dan aktivitas. Dengan menggunakan *UML*, para pengembang perangkat lunak dapat membuat diagram yang terstruktur dan mudah dimengerti, membantu dalam pemahaman menyeluruh terhadap sistem, pemodelan proses bisnis, identifikasi kebutuhan

sistem, serta analisis dan perancangan sistem. *UML* mendukung berbagai jenis diagram yang umum digunakan dalam proses rekayasa perangkat lunak.

### 2.9.1 Use Case Diagram

*Use case diagram* adalah representasi visual yang digunakan untuk menggambarkan interaksi antara pengguna atau sistem eksternal dengan sistem yang sedang dikembangkan (Ahdan dkk., 2019). Diagram ini membantu *software developer* dalam menganalisis kebutuhan fungsionalitas sistem dari perspektif pengguna dan objek yang terlibat. Selama proses analisis dan siklus hidup sistem, use case diagram dapat menggambarkan apa yang diinginkan oleh pengguna terhadap apa yang dapat dilakukan oleh sistem. Simbol-simbol yang ada pada *use case diagram* dapat dilihat pada Tabel 2.1.

**Tabel 2.1** Simbol-simbol *Use Case Diagram*

Nama	Simbol	Penjelasan
System		Boundary antara sistem dengan pengguna sistem.
Use case		Mewakili satu skenario yang dilakukan oleh actor atau sistem dan membantu memodelkan fungsionalitas sistem.
Actor		Mewakili pengguna nyata atau sistem eksternal yang terlibat interaksi dengan sistem. <i>Actor</i> dapat berupa manusia, perangkat keras, perangkat lunak dan sistem eksternal lainnya.

Lanjutan Tabel 2.1 Simbol-simbol *Use Case Diagram*

Nama	Simbol	Penjelasan
<i>Association</i>	———	Menggambarkan keterlibatan <i>actor</i> dalam aksi atau <i>use case</i> yang didefinisikan.
<i>Include</i>	-----> <<include>>	Menggambarkan dimana suatu <i>use case</i> meliputi atau membutuhkan fungsionalitas dari <i>use case</i> lainnya.
<i>Generalization</i>	———>	Menggambarkan adanya kaitan hirarki antar <i>use case</i> , dimana suatu <i>use case</i> mewarisi sifat dan perilaku <i>use case</i> lainnya.
<i>Extend</i>	-----> <<extend>>	Menggambarkan bahwa suatu <i>use case</i> bisa diperluas dengan tambahan fungsi yang diaktifkan oleh <i>use case</i> lainnya hal ini bersifat opsional.

### 2.9.2 *Activity Diagram*

*Activity diagram* adalah diagram yang berfungsi memvisualisasikan aliran kerja atau aktivitas dalam sebuah proses sistem (Simangunsong & Voutama, 2023).

Pada diagram ini menampilkan serangkaian aktivitas, tindakan, keputusan, dan kontrol aliran yang terjadi dalam proses sistem.

*Activity diagram* dapat terdiri dari cabang dan pengelompokan sistem kontrol menjadi cabang yang sama, secara bersamaan mewakili aktivitas yang dapat dikerjakan secara bersamaan oleh objek yang berbeda. *Activity diagram* memiliki simbol-simbol yang dapat dilihat pada Tabel 2.2.

**Tabel 2.2** Simbol-simbol *Activity Diagram*

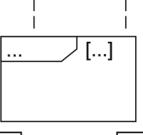
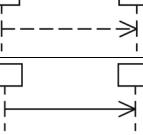
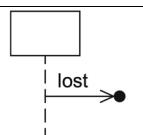
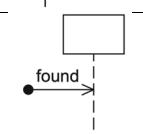
Nama	Simbol	Penjelasan
<i>Partition</i>		Pemisah vertikal dalam diagram yang berfungsi untuk mengorganisir aktivitas-aktivitas dalam diagram berdasarkan unit yang bertanggung jawab.
<i>Activities</i>		Memvisualisasikan tindakan yang terjadi dalam sistem.
<i>Initial node</i>		Menandakan titik mulai jalannya alur sistem.
<i>Activity final node</i>		Menandakan titik akhir dari jalannya alur sistem lainnya.
<i>Flow final node</i>		Menandakan akhir dari satu jalur eksekusi suatu aktivitas.
<i>Decision Mode</i>		Pemisahan satu jalur eksekusi menjadi dua atau lebih jalur eksekusi alternatif.
<i>Merge node</i>		Penggabungan dua atau lebih jalur eksekusi alternatif menjadi satu jalur eksekusi.
<i>Parallelization node</i>		Pemisahan satu jalur eksekusi menjadi dua atau lebih jalur eksekusi secara bersamaan.
<i>Synchronization node</i>		Penggabungan dua atau lebih jalur eksekusi bersamaan menjadi satu jalur eksekusi.
<i>Edge</i>		Koneksi antar node suatu aktivitas.

### 2.9.3 Sequence Diagram

*Sequence diagram* merupakan diagram yang digunakan untuk memvisualisasikan interaksi antara objek-objek dalam sistem (Winata & Setiawan, 2013), dimana pesan yang dikirimkan antar objek dan urutan waktu dapat diketahui sehingga mempermudah pemahaman terhadap analisis sistem yang berjalan. Diagram ini menampilkan interaksi dalam bagan dua dimensi, dimensi vertikal

melambangkan sumbu waktu berjalan dari atas ke bawah dan dimensi horizontal melambangkan fungsi pengklasifikasi yang mewakili objek individu. *Sequence diagram* memiliki simbol-simbol yang dapat dilihat pada Tabel 2.3.

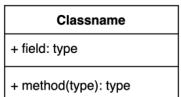
**Tabel 2.3** Simbol-simbol *Sequence Diagram*

Nama	Simbol	Penjelasan
Lifeline		Menggambarkan entitas yang terlibat.
Combined Fragment		Memvisualisasikan kontrol konstruksi.
Synchronous Message		Pengirim menunggu pesan balasan.
Response Message		Respons terhadap pesan sinkron.
Asynchronous Message		Pengirim melanjutkan pekerjaannya sendiri setelah mengirim pesan asinkron.
Lost Message		Pesan ke penerima yang tidak dikenal.
Found Message		Menggambarkan pesan dari pengirim yang tidak dikenal.

#### 2.9.4 Class Diagram

*Class diagram* merupakan jenis diagram yang memiliki struktur statis yang menampilkan struktur sistem dengan menuliskan kelas sistem, atribut yang dimiliki, metode serta hubungan antar objek (Feby Prasetya & Lestari Dewi Putri, 2022). *Class diagram* memiliki simbol-simbol yang dapat dilihat pada Tabel 2.4.

**Tabel 2.4 Simbol-simbol Class Diagram**

Nama	Simbol	Penjelasan
Class		Menggambarkan entitas eksternal yang memulai interaksi dengan objek-objek dalam sistem, dapat berupa manusia, sistem eksternal dan komponen lainnya.
Link		Menggambarkan hubungan asosiasi antara kelas-kelas yang terlibat.
Object		Instance dari sebuah kelas.
Strong Aggregation (Composition)		Relasi yang menggambarkan hubungan antara kelas induk dan anak, dimana kelas anak merupakan bagian terpenting dari kelas induk yang tidak bisa berdiri sendiri.
Shared Aggregation		Relasi yang menggambarkan hubungan antara kelas induk dan anak, yang dimana kelas anak tidak memiliki ketergantungan dengan kelas induk sehingga dapat berdiri secara sendiri.
Generalization		Relasi yang menggambarkan bahwa sebuah kelas dapat mewarisi atribut dan metode class lain.

## 2.10 Black Box Testing

*Black box testing* adalah metode pengujian perangkat lunak di mana pengujian dilakukan tanpa memerhatikan atau memiliki pengetahuan tentang struktur *internal* atau logika implementasi dari sistem yang diuji (Setiana dkk., 2024). Fokus utama dari *black box testing* adalah pada fungsionalitas eksternal sistem dan kemampuannya untuk memenuhi persyaratan spesifikasi yang telah ditetapkan. Dalam hal ini, pengujian dilakukan sebagai input ke sistem dan hasil

keluarannya dievaluasi tanpa kebutuhan untuk mengetahui bagaimana sistem mencapai hasil tersebut.

### **2.11 *User Acceptance Testing***

*User acceptance testing (UAT)* merupakan pengujian yang dilakukan oleh pengguna akhir yang langsung berinteraksi dengan sistem (Saputra dkk., 2022). Tujuannya adalah untuk memverifikasi apakah fungsi-fungsi yang ada telah berjalan sesuai dengan kebutuhan atau fungsinya dan memastikan bahwa sistem dapat membantu para pengguna dengan efektif. *UAT* dilakukan pada tahap akhir pengembangan perangkat lunak untuk memastikan bahwa sistem tidak hanya memenuhi spesifikasi teknis, tetapi juga memenuhi kebutuhan sebenarnya dari pengguna.

Pengujian *UAT* dilakukan dengan cara menyebarluaskan kuesioner kepada pengguna aplikasi *auto grader*. Kuesioner tersebut bertujuan untuk mengetahui sejauh mana tingkat penerimaan aplikasi *auto grader*. Hasil persentase dari tiap pertanyaan yang diberikan kepada responden memiliki 5 skala menggunakan skala *Likert*. Skala *Likert* merujuk pada sebuah metode pengukuran yang digunakan untuk mengevaluasi pandangan individu terhadap suatu objek kajiannya melalui penggunaan kuesioner. Pendekatan ini bertujuan untuk menilai sejauh mana sikap individu terhadap objek yang dianalisis. Pada skala *likert* ini kuesioner yang diberikan memberikan skor pada setiap jawaban yaitu 1-5. skor 1 (satu) untuk pendapat atau jawaban sangat tidak setuju sedangkan skor 5 (lima) untuk pendapat atau jawaban sangat setuju. Skala *likert* dapat dilihat pada Tabel 2.5.

**Tabel 2.5** Skala Likert

<b>Bobot</b>	<b>Keterangan</b>
1	Sangat tidak setuju
2	Tidak setuju
3	Cukup
4	Setuju
5	Sangat Setuju

Data bobot kuesioner yang didapat kemudian digunakan untuk menghitung nilai rata-rata. Berikut ini merupakan rumus nilai rata-rata.

$$\text{Nilai Rata Rata} = \frac{\text{Jumlah Bobot Nilai Responden}}{\text{Total Responden}} \dots \quad (2.1)$$

Setelah mendapatkan nilai rata-rata, maka dilanjutkan dengan perhitungan persentase pertanyaan yang dilakukan untuk mendapatkan hasil kualitas sistem untuk layak untuk digunakan bagi pengguna.

Berdasarkan hasil total persentase, dapat ditentukan keterangan dari persentase yang diperoleh dengan mengacu pada kriteria interpretasi skor yang terdapat pada Tabel 2.6.

**Tabel 2.6** Kriteria Interpretasi Skor

<b>Presentase</b>	<b>Keterangan</b>
0% - 20%	Sangat kurang baik
21% - 40%	Kurang baik
41% - 60%	Cukup Baik
61% - 80%	Baik
81% - 100%	Sangat Baik

## **2.12 Penelitian Terdahulu**

Penelitian terdahulu oleh (Wang dkk., 2020) berjudul *Combining Dynamic and Static Analysis for Automated Grading SQL Statements* mengusulkan pendekatan penilaian otomatis untuk pernyataan *SQL*. Pendekatan ini menggabungkan analisis dinamis dan statis. Pertama, analisis dinamis digunakan untuk menjalankan pernyataan *SQL* mahasiswa, diikuti oleh analisis statis untuk mengevaluasi strukturnya tanpa menjalankannya. Analisis statis ini fokus pada fitur-fitur khusus dari pernyataan *SQL* yang dapat menghasilkan skor terperinci.

Penelitian ini juga mencatat bahwa sebagian besar sistem penilaian otomatis di platform seperti *CodeChef*, *CodeForces*, dan *Hackerrank* menggunakan analisis dinamis. Hasil penelitian menyimpulkan bahwa berbagai pendekatan dalam pengorekan memiliki kelebihan dan kekurangan masing-masing, dan menggabungkannya dapat meningkatkan kualitas penilaian.

Dalam penelitian lain yang berjudul “Rancang Bangun Sistem MyITS Dorm Menggunakan Metode *Domain Driven Design* dan *Onion Architecture*” merancang sistem untuk mengelola asrama mahasiswa. penelitian tersebut menggunakan metode *Domain Driven Design* dan *Onion Architecture* untuk menghasilkan system yang terstruktur dan efisien. Hasilnya adalah implementasi system MyITS Dorm yang memenuhi kebutuhan bisnis dengan baik. Dalam penelitian yang berjudul “Identifikasi Kandidat *Microservices* dengan Analisis *Domain Driven Design*” yang dilakukan oleh (Juliawan dkk., 2021), penulis menggunakan analisis *Domain Driven Design* untuk mengidentifikasi kandidat microservices berbasis *RESTful API* dalam rangka migrasi Sistem Informasi Dosen (SIMDOSEN) di Universitas

Udayana dari arsitektur *monolithic* ke arsitektur *microservices*. Hasil penelitian ini menunjukkan proses pengembangan *microservices* yang sistematis dan memberikan dampak positif pada kemajuan arsitektur IT di Universitas Udayana.

## **BAB III**

### **METODE PENELITIAN**

#### **3.1 Teknik Pengumpulan Data**

Pada tahap penelitian ini dilakukannya pengumpulan data yang dibutuhkan agar penelitian dapat berjalan dengan lancar. Pengumpulan data tersebut dilakukan dengan tujuan mendapatkan pemahaman terkait *auto grader* dan menganalisa kebutuhan aplikasi yang akan dibangun. Adapun teknik pengumpulan data yang digunakan sebagai berikut:

##### 1. Studi Literatur

Studi literatur merupakan suatu pendekatan penelitian yang bertujuan untuk menginvestigasi, menganalisis, dan mengintegrasikan informasi yang terdapat dalam literatur ilmiah yang berkaitan dengan subjek penelitian. Pendekatan dilakukan dengan cara membaca, memahami dan mencatat literatur ilmiah yang bersumber dari jurnal, buku, *ebook* dan pencarian internet yang berhubungan dengan *auto grader* dan *domain driven design*.

##### 2. Wawancara

Wawancara merupakan pendekatan yang melibatkan interaksi langsung antara peneliti dan narasumber dengan tujuan memperoleh *requirement* atau kebutuhan aplikasi *auto grader*. Dalam pendekatan ini wawancara dilakukan dengan seorang dosen di Jurusan Ilmu Komputer Universitas Riau.

### **3.2 Peralatan yang Digunakan**

Terdapat beberapa peralatan yang berfungsi sebagai alat pendukung dalam proses pembuatan aplikasi dan melakukan penelitian diantaranya:

1. Perangkat Keras

Perangkat keras yang digunakan selama penelitian ini adalah:

- a. Laptop MacBook Air 2020 dengan spesifikasi *processor Apple M1 8-core CPU, 7-core GPU, 16-core Neural Engine, SSD 256 GB, RAM 8 GB.*

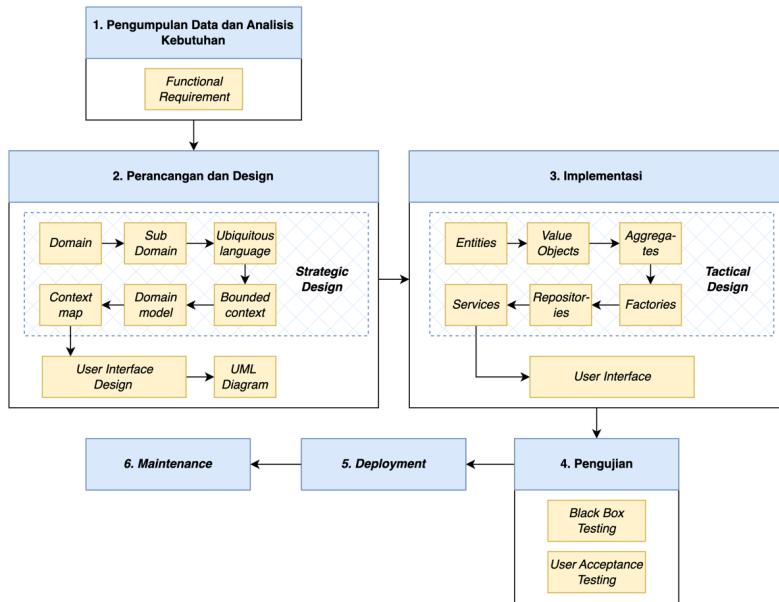
2. Perangkat Lunak

Beberapa perangkat lunak yang digunakan selama penelitian ini adalah:

- a. Sistem Operasi *MacOS Sonoma 14.5*
- b. *Goland* dan *PHPStorm IDE*
- c. *Java SE 11*
- d. *Apache Maven*
- e. *MongoDB*
- f. *Redis*
- g. *Mendeley*
- h. *Microsoft Office 365*
- i. *Neva Cloud*

### **3.3 Langkah-Langkah Penyelesaian**

Urutan Langkah-langkah penyelesaian yang dilakukan dalam penelitian ini dapat dilihat pada Gambar 3.1.



**Gambar 3.1** Bagan Tahapan Penelitian

1. Pengumpulan data dan analisis kebutuhan

Tahapan awal yang harus dilakukan adalah mengumpulkan data-data yang dibutuhkan. Penelitian ini menggunakan dua jenis data yaitu data primer berupa hasil wawancara untuk mendapatkan gambaran dan kebutuhan mengenai sistem *auto grader* yang akan dibangun dan data sekunder berupa studi literatur.

2. Perancangan dan desain

Setelah kebutuhan aplikasi *auto grader* didefinisikan dengan baik maka langkah selanjutnya adalah merancang dan membangun desain aplikasi. Tahapan ini akan menerapkan dua pilar *domain driven design* yaitu *strategic*

*design* dan *ubiquitous language* yang akan menghasilkan *sub-domain*, *ubiquitous language*, *bounded context*, *domain model*, dan *context map*. Kemudian alur aplikasi digambarkan menggunakan *UML* yaitu *use case diagram*, *activity diagram*, *sequence diagram*, dan *class diagram*. Terakhir Membuat *user interface design* untuk memberikan gambaran tampilan aplikasi yang akan dibangun serta memberikan kemudahan pada proses implementasi.

### 3. Implementasi

Tahapan ini akan mengimplementasikan rancangan dan *design* yang telah dibuat sebelumnya ke dalam bahasa pemrograman dan menerapkan *tactical design* yang mencakup *entities*, *aggregates*, *value objects*, *factories*, *repositories*, dan *services*.

### 4. Pengujian

Proses pengujian aplikasi dilakukan menggunakan metode pengujian *black box* dan *user acceptance testing*. Pengujian menggunakan *black box testing* membantu untuk menilai apakah aplikasi berjalan sesuai standar tanpa perlu mengetahui sumber kode dan algoritma yang ada didalamnya. Sementara itu, *user acceptance testing* digunakan untuk mengetahui tanggapan pengguna atau responden terhadap sistem yang telah dibangun dengan menggunakan kuesioner.

### 5. Deployment

Setelah tahap pengujian berhasil dilakukan dan aplikasi dianggap siap untuk digunakan, langkah selanjutnya adalah melakukan *deployment*. Pada tahap

*deployment*, aplikasi *auto grader* akan diimplementasikan dan diaktifkan di lingkungan produksi. Proses ini melibatkan instalasi perangkat lunak, konfigurasi, dan penyesuaian agar aplikasi dapat berjalan dengan lancar. *Deployment* merupakan langkah kritis karena aplikasi harus dapat beroperasi secara optimal dan dapat diakses oleh pengguna sesuai dengan kebutuhan.

#### 6. *Maintainance*

Setelah aplikasi di *deploy*, tugas utama tidak berhenti di situ. Tahapan *maintainance* (pemeliharaan) menjadi penting untuk memastikan bahwa aplikasi tetap berfungsi secara optimal dan dapat menanggapi perubahan yang mungkin terjadi di lingkungan atau kebutuhan pengguna. *Maintenance* melibatkan pemantauan kinerja, penanganan *bug*, pembaruan perangkat lunak, dan peningkatan fungsionalitas sesuai dengan *feedback* pengguna.

## **BAB IV**

### **HASIL DAN PEMBAHASAN**

#### **4.1 Pengumpulan Data dan Analisis Kebutuhan**

Pengumpulan data dilakukan dengan tujuan agar memperoleh data yang dibutuhkan dalam melakukan penelitian sehingga penelitian dapat berjalan semestinya. Terdapat dua jenis data yang dikumpulkan dalam penelitian ini yaitu:

##### 1. Data Primer

Data primer adalah informasi yang diperoleh secara langsung dari sumber terpercaya atau pakar dalam bidang tertentu. Dalam penelitian ini, data primer diperoleh melalui wawancara langsung dengan Bapak Al Aminuddin, ST., M.Sc., selaku dosen di Jurusan Ilmu Komputer. Beliau merupakan salah satu dosen pengampu mata kuliah Konsep Pemrograman, Pemrograman Berorientasi Objek, dan Algoritma Struktur Data. Mata kuliah tersebut sering mengadakan praktikum penulisan kode program, sehingga beliau tepat untuk menjadi *domain expert* untuk aplikasi *auto grader*. Data yang didapatkan disusun menjadi daftar kebutuhan atau *requirement* dalam bentuk *functional requirements* dari aplikasi *auto grader*. *Functional requirements* adalah spesifikasi rinci tentang fitur perangkat lunak yang menjelaskan apa yang harus dilakukan oleh sistem untuk memenuhi kebutuhan pengguna. Daftar *functional requirements* dapat dilihat pada Tabel 4.1.

**Tabel 4.1** Daftar Functional Requirements Auto Grader

No	Aktor	Fitur	Deskripsi
1.	User	Login	Sistem harus menyediakan fitur bagi pengguna untuk masuk ke aplikasi menggunakan akun <i>google</i> .
2.	User	Bergabung menggunakan kode kelas	Sistem harus menyediakan fitur bagi pengguna untuk bergabung dengan kelas menggunakan kode kelas yang valid.
3.	User	Membuat kelas	Sistem harus menyediakan fitur bagi pengguna untuk membuat kelas baru.
4.	Class Member	Melihat detail tugas	Sistem harus menyediakan fitur bagi anggota kelas untuk melihat detail dari tugas yg telah diberikan.
5.	Class Member	Melihat riwayat submission	Sistem harus menyediakan fitur bagi anggota kelas untuk melihat riwayat submission yg telah dikumpulkan.
6.	Class Member	Melihat leaderboard	Sistem harus menyediakan fitur bagi anggota kelas untuk bisa melihat peringkat dari pengumpulan tugas.
7.	Class Member	Melihat anggota kelas	Sistem harus menyediakan fitur bagi anggota kelas untuk bisa melihat daftar anggota dari setiap kelas yg mereka ikuti.
8.	Student	Mengumpulkan tugas	Sistem harus menyediakan fitur bagi siswa untuk bisa mengumpulkan atau <i>submit</i> tugas yang telah dibuat.
9.	Teacher	Mengelola tugas	Sistem harus menyediakan fitur bagi pengajar untuk bisa mengelola tugas seperti membuat tugas baru, mengedit tugas, menghapus, dan mengunduh rekap nilai tugas.
10.	Class Owner	Mengelola kelas	Sistem harus menyediakan fitur bagi pemilik kelas untuk bisa mengelola kelas seperti mengedit dan menghapus kelas.
11.	Class Owner	Mengelola anggota kelas	Sistem harus menyediakan fitur bagi pemilik kelas untuk bisa mengelola anggota kelas seperti menghapus dan menambahkan anggota baru.

## 2. Data Sekunder

Data sekunder adalah informasi yang telah tersedia dalam berbagai format, seperti laporan riset sebelumnya, data numerik, karya akademis, dokumen

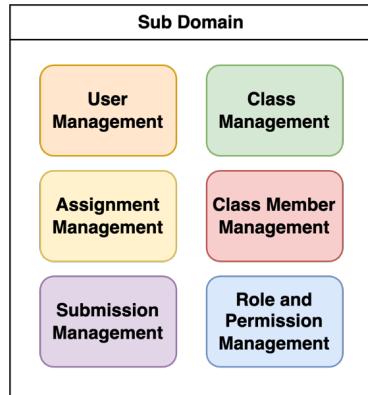
resmi pemerintah, dan sumber-sumber lainnya. Dalam penelitian ini, data sekunder diperoleh melalui studi literatur, yaitu dengan mencari informasi terkait permasalahan penelitian dari artikel ilmiah seperti buku dan jurnal. Artikel ilmiah dan buku yang relevan dengan penelitian ini membahas *auto grading system* dan penerapan *domain driven design* dalam pengembangan perangkat lunak.

#### 4.2 Perancangan dan *Design* Perangkat Lunak

Setelah membuat *requirement* berupa *user stories*, tahapan berikutnya yaitu menentukan *sub-domain*, daftar istilah, *bounded context*, *context map*, *user interface design*, dan diagram *UML* dari aplikasi yang akan dikembangkan.

##### 4.2.1 *Domain* dan *Sub-Domain*

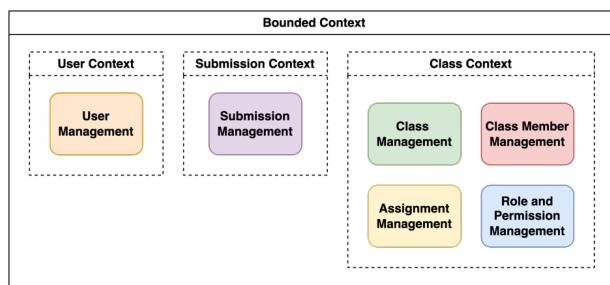
*Domain* merujuk pada cakupan keseluruhan dari aktivitas utama atau layanan yang diberikan kepada pengguna. Domain dari aplikasi *auto grader* adalah “*Auto Grading* atau Penilaian Otomatis”. Sedangkan *Subdomain* merujuk pada cara untuk mengatasi kompleksitas dengan memecah masalah yang kompleks menjadi masalah yang lebih mudah dikelola. Terdapat 6 *subdomain* dari *domain auto grading*. *Subdomain* dapat dilihat pada Gambar 4.1.



**Gambar 4.1 Sub Domain Auto Grader**

#### 4.2.2 *Bounded Context*

*Bounded context* adalah batasan yang digunakan untuk membatasi cakupan dari *domain model* yang dibuat dalam pengembangan perangkat lunak. Teknik yang digunakan untuk menentukan *bounded context* pada aplikasi *auto grader* dilihat berdasarkan kapabilitas nya. *Bounded context* dapat dilihat pada Gambar 4.2.



**Gambar 4.2 Bounded Context Auto Grader**

#### 4.2.3 Daftar Istilah

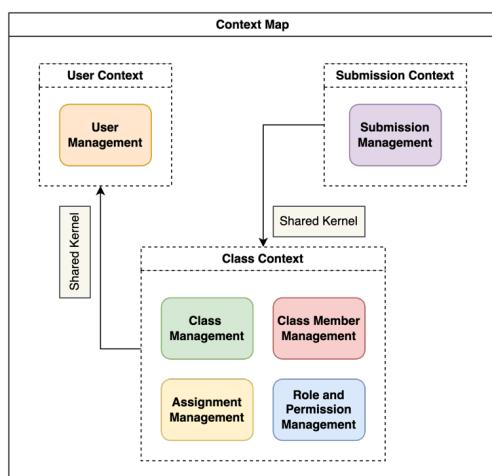
Daftar istilah digunakan untuk media komunikasi antara pengembang atau *technical expert* dengan *domain expert*. Daftar istilah dapat dilihat pada Tabel 4.2.

**Tabel 4.2** Daftar Istilah atau *Ubiquitous Language*

No	Istilah	Definisi
1.	<i>User</i> (Pengguna)	Individu yang telah memiliki akses ke sistem
2.	<i>Class</i> (Kelas)	Kelas merujuk pada ruang pembelajaran virtual di mana pengajar dapat mengelola tugas.
3.	<i>Assignment</i> (Tugas)	Tugas atau pekerjaan yang diberikan oleh pengajar kepada siswa untuk diselesaikan dan diunggah ke dalam sistem sebagai bagian dari evaluasi pembelajaran.
4.	<i>Submission</i> (Pengumpulan)	Proses mengirimkan atau mengunggah tugas atau pekerjaan oleh siswa ke dalam sistem sebagai respons terhadap suatu tugas.
5.	<i>Owner</i> (Pemilik)	Individu yang memiliki tanggung jawab penuh atas suatu kelas termasuk informasi kelas, manajemen tugas, pengajar, dan siswa.
6.	<i>Teacher</i> (Pengajar)	Pengguna yang terdaftar dalam suatu kelas yang memiliki peran untuk mengelola tugas.
7.	<i>Student</i> (Siswa)	Pengguna yang terdaftar dalam suatu kelas sebagai seorang siswa.
8.	<i>Leaderboard</i> (Peringkat atau Skor)	Papan peringkat menampilkan skor siswa pada suatu tugas tertentu.
9.	<i>Log</i> (Catatan)	Keterangan dari submission siswa yang berisi informasi jawaban yang benar dan salah.
10.	<i>Being Graded</i> (Sedang Dinilai)	Status yang menunjukkan bahwa <i>submission</i> sedang dalam proses penilaian oleh sistem.
11.	<i>Successfully Graded</i> (Berhasil Dinilai)	Status yang menunjukkan bahwa <i>submission</i> telah berhasil dinilai.
13.	<i>Failed To Grade</i> (Gagal Dinilai)	Status yang menunjukkan bahwa <i>submission</i> gagal dinilai atau tidak berhasil melalui proses penilaian, karena <i>submission</i> yang tidak sesuai dengan ketentuan atau masalah teknis pada sistem.

#### 4.2.4 Context Map

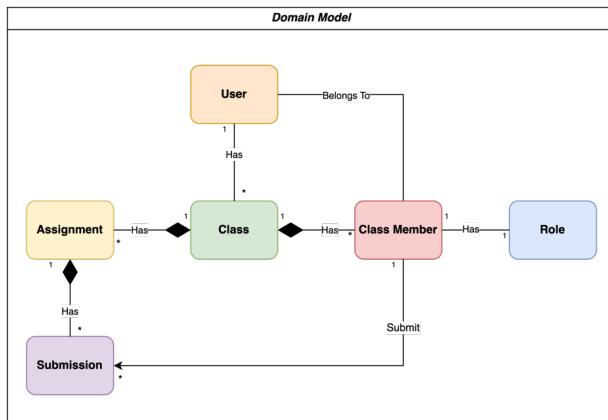
*Context map* merupakan representasi visual dari komunikasi antara berbagai *bounded context*. *Context map* pada aplikasi *auto grader* menggunakan *shared kernel*. *Shared kernel* digunakan karena *model* atau artefak dari masing-masing *bounded context* yang berhubungan digunakan secara bersamaan. *Bounded context* dapat dilihat pada Gambar 4.3.



Gambar 4.3 *Context Map*

#### 4.2.5 Domain Model

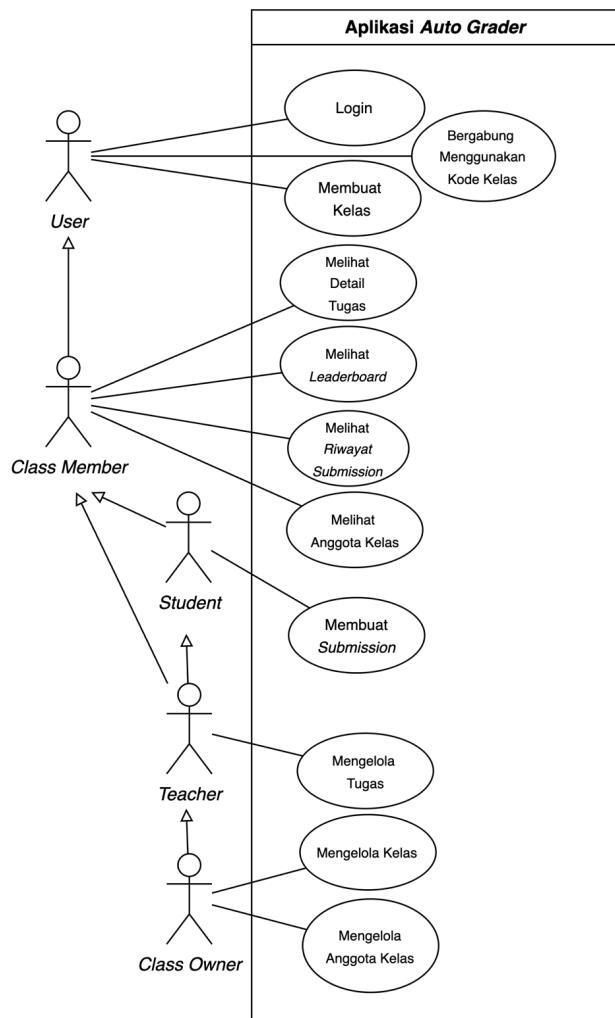
*Domain model* merupakan gambaran struktural dari perangkat lunak yang dibangun. *Model* ini dibangun untuk menyamakan persepsi antara *technical expert* dengan *domain expert*. *Domain model* dapat memberikan *preview* mengenai *object domain* yang nantinya akan diimplementasikan ke dalam bahasa pemrograman. *Domain model* dapat dilihat pada Gambar 4.4.



**Gambar 4.4 Domain Model**

#### 4.2.6 Use Case Diagram

*Use case diagram* menggambarkan interaksi apa saja yang dapat dilakukan antara pengguna dengan aplikasi. Ketika maksud di balik *use case* dan interaksinya tidak jelas, maka harus mendeskripsikan *use case* tersebut menggunakan *use case specification*. *Use case diagram* yang telah dirancang dapat dilihat pada Gambar 4.5.



**Gambar 4.5 Use Case Diagram**

Berikut merupakan spesifikasi dari masing-masing *use case* diatas:

**Tabel 4.3** Deskripsi *Use Case Login*

<i>Name:</i>	Login
<i>Short Description:</i>	Pengguna dapat masuk ke sistem menggunakan akun <i>Google</i> mereka melalui proses autentikasi <i>OAuth</i> .
<i>Precondition:</i>	<ol style="list-style-type: none"> <li>1. Pengguna memiliki akun <i>Google</i> yang aktif.</li> <li>2. Sistem telah terintegrasi dengan layanan <i>OAuth Google</i> untuk autentikasi.</li> <li>3. Browser pengguna harus mendukung <i>JavaScript</i> dan <i>cookie</i>.</li> </ol>
<i>Postcondition:</i>	<p>Berhasil: Pengguna berhasil masuk ke sistem dan diarahkan ke halaman utama atau dashboard.</p> <p>Gagal: Pengguna tetap berada di halaman login dengan pesan kesalahan yang sesuai atau diarahkan kembali dari <i>Google</i> dengan notifikasi kesalahan.</p>
<i>Error situations:</i>	<ol style="list-style-type: none"> <li>1. Pengguna memiliki akun <i>Google</i> yang aktif.</li> <li>2. Terjadi kesalahan pada layanan <i>Google OAuth</i></li> </ol>
<i>System state in the event of an error:</i>	<ol style="list-style-type: none"> <li>1. Sistem menampilkan pesan kesalahan yang sesuai di halaman <i>login</i>.</li> <li>2. Kredensial yang dimasukkan tidak disimpan oleh sistem.</li> </ol>
<i>Actors:</i>	Pengguna
<i>Trigger:</i>	Pengguna memilih opsi " <i>Login with Google</i> " pada halaman <i>login</i> .
<i>Standard process:</i>	<ol style="list-style-type: none"> <li>1. Pengguna membuka halaman <i>login</i>.</li> <li>2. Sistem menampilkan opsi "<i>Login dengan Google</i>".</li> <li>3. Pengguna menekan tombol "<i>Login dengan Google</i>".</li> <li>4. Sistem mengarahkan pengguna ke halaman autentikasi <i>Google</i>.</li> <li>5. Pengguna memasukkan kredensial <i>Google</i> dan memberikan izin akses.</li> <li>6. <i>Google</i> memverifikasi kredensial dan izin akses, kemudian mengarahkan kembali pengguna ke sistem dengan token autentikasi.</li> <li>7. Sistem menerima token dari <i>Google</i> dan memverifikasi token tersebut.</li> <li>8. Jika token valid, sistem mengarahkan pengguna ke halaman utama atau dashboard.</li> </ol>
<i>Alternative processes:</i>	<ol style="list-style-type: none"> <li>1. <i>Google</i> menampilkan pesan kesalahan bahwa kredensial tidak valid atau akun tidak aktif.</li> <li>2. Pengguna dapat mencoba kembali atau menghubungi dukungan <i>Google</i>.</li> </ol>

**Tabel 4.4** Deskripsi Use Case Membuat Kelas

<i>Name:</i>	Membuat Kelas
<i>Short Description:</i>	Pengguna membuat kelas baru dalam sistem untuk memfasilitasi pembelajaran.
<i>Precondition:</i>	1. Pengguna telah terdaftar dan masuk ke dalam sistem.
<i>Postcondition:</i>	Berhasil: Kelas baru berhasil dibuat dan tersedia dalam daftar kelas. Gagal: Kelas tidak dibuat, dan pengguna tetap berada di halaman pembuatan kelas dengan pesan kesalahan yang sesuai.
<i>Error situations:</i>	1. Pengguna tidak mengisi semua inputan yang diperlukan. 2. Format data yang dimasukkan tidak valid 3. Terjadi masalah pada sistem atau server saat menyimpan data kelas.
<i>System state in the event of an error:</i>	1. Sistem menampilkan pesan kesalahan yang sesuai di halaman pembuatan kelas. 2. Data yang dimasukkan oleh pengguna tidak disimpan dan tetap ada di form untuk koreksi.
<i>Actors:</i>	Pengguna
<i>Trigger:</i>	Pengguna memilih opsi "Buat Kelas Baru" dari dashboard atau menu utama.
<i>Standard process:</i>	1. Pengguna membuka halaman pembuatan kelas dengan memilih opsi "Buat Kelas Baru". 2. Sistem menampilkan form pembuatan kelas dengan berbagai <i>input</i> yang diperlukan 3. Pengguna mengisi semua bidang yang diperlukan pada form. 4. Instruktur menekan tombol "Simpan" atau "Buat Kelas". 5. Sistem memverifikasi bahwa semua inputan yang diperlukan telah diisi dengan data yang <i>valid</i> . 6. Sistem menyimpan data kelas baru ke dalam database. 7. Sistem mengarahkan instruktur ke halaman konfirmasi atau halaman detail kelas yang baru dibuat. 8. Kelas baru muncul dalam daftar kelas yang tersedia.
<i>Alternative processes:</i>	1. Sistem menampilkan pesan kesalahan yang menunjukkan bahwa ada <i>input form</i> yang belum diisi. 2. Pengguna melengkapi <i>input form</i> yang diperlukan dan melanjutkan dari langkah 4 Proses Standar.

**Tabel 4.5** Deskripsi *Use Case* Bergabung Menggunakan Kode Kelas

<i>Name:</i>	Bergabung Menggunakan Kode Kelas
<i>Short Description:</i>	Siswa dapat bergabung dengan kelas di aplikasi menggunakan kode kelas yang diberikan oleh instruktur.
<i>Precondition:</i>	1. Pengguna telah terdaftar dan masuk ke dalam sistem. 2. Siswa memiliki kode kelas yang valid dari instruktur.
<i>Postcondition:</i>	Berhasil: Siswa berhasil bergabung dengan kelas dan dapat mengakses konten kelas. Gagal: Siswa tetap berada di halaman bergabung dengan pesan kesalahan yang sesuai.
<i>Error situations:</i>	1. Kode kelas yang dimasukkan tidak valid atau salah. 2. Terjadi masalah pada sistem atau server saat memverifikasi kode kelas. 3. Siswa sudah terdaftar dalam kelas yang sama
<i>System state in the event of an error:</i>	1. Sistem menampilkan pesan kesalahan yang sesuai di halaman bergabung. 2. Data yang dimasukkan oleh siswa tidak disimpan.
<i>Actors:</i>	Pengguna
<i>Trigger:</i>	Pengguna memilih opsi "Bergabung dengan Kelas" dan memasukkan kode kelas.
<i>Standard process:</i>	1. Siswa membuka halaman "Bergabung dengan Kelas". 2. Sistem menampilkan form untuk memasukkan kode kelas. 3. Siswa memasukkan kode kelas yang diberikan oleh instruktur. 4. Siswa menekan tombol "Bergabung". 5. Sistem memverifikasi kode kelas yang dimasukkan. 6. Jika kode kelas valid, sistem menambahkan siswa ke kelas tersebut. 7. Sistem mengarahkan siswa ke halaman utama kelas atau dashboard kelas. 8. Kelas baru muncul dalam daftar kelas yang diikuti oleh siswa
<i>Alternative processes:</i>	1. Sistem menampilkan pesan kesalahan bahwa kode kelas tidak valid. 2. Siswa dapat mengoreksi kode kelas yang dimasukkan dan melanjutkan dari langkah 3 Proses Standar.

**Tabel 4.6** Deskripsi Use Case Melihat Detail Tugas

<i>Name:</i>	Melihat Detail Tugas
<i>Short Description:</i>	Anggota kelas dapat melihat informasi lengkap mengenai tugas yang diberikan oleh instruktur, termasuk deskripsi, tanggal jatuh tempo, dan kriteria penilaian.
<i>Precondition:</i>	1. Anggota kelas telah terdaftar dan masuk ke dalam sistem. 2. Anggota kelas sudah terdaftar di kelas yang terkait dengan tugas tersebut.
<i>Postcondition:</i>	Berhasil: Anggota kelas berhasil melihat detail tugas yang diinginkan. Gagal: Anggota kelas tetap berada di halaman sebelumnya dengan pesan kesalahan yang sesuai.
<i>Error situations:</i>	1. Tugas tidak ditemukan atau tidak tersedia. 2. Terjadi masalah pada sistem atau server saat mengambil informasi tugas.
<i>System state in the event of an error:</i>	Sistem menampilkan pesan kesalahan yang sesuai di halaman bergabung.
<i>Actors:</i>	Anggota kelas
<i>Trigger:</i>	Anggota kelas memilih opsi untuk melihat detail tugas dari daftar tugas yang tersedia dalam kelas.
<i>Standard process:</i>	1. Anggota kelas membuka halaman daftar tugas dalam kelas tertentu. 2. Sistem menampilkan daftar tugas yang tersedia untuk kelas tersebut. 3. Siswa memilih tugas yang diinginkan untuk melihat detailnya. 4. Sistem mengambil informasi lengkap mengenai tugas dari database. 5. Sistem menampilkan detail tugas, termasuk deskripsi, tanggal jatuh tempo, dan kriteria penilaian. 6. Anggota kelas dapat melihat semua informasi terkait tugas tersebut.
<i>Alternative processes:</i>	1. Sistem menampilkan pesan kesalahan bahwa tugas tidak ditemukan. 2. Siswa kembali ke halaman daftar tugas dan dapat memilih tugas lain.

**Tabel 4.7** Deskripsi Use Case Melihat Leaderboard

<i>Name:</i>	Melihat Leaderboard
<i>Short Description:</i>	Anggota kelas dapat melihat peringkat berdasarkan skor tugas tertentu yang diberikan oleh instruktur.
<i>Precondition:</i>	1. Anggota kelas telah terdaftar dan masuk ke dalam sistem. 2. <i>Submission</i> telah dinilai oleh sistem.
<i>Postcondition:</i>	Berhasil: Anggota kelas berhasil melihat <i>leaderboard</i> tugas yang diinginkan. Gagal: Anggota kelas tetap berada di halaman sebelumnya dengan pesan kesalahan yang sesuai.
<i>Error situations:</i>	1. Tugas atau leaderboard tidak ditemukan atau tidak tersedia. 2. Terjadi masalah pada sistem atau server saat mengambil informasi tugas.
<i>System state in the event of an error:</i>	Sistem menampilkan pesan kesalahan yang sesuai di halaman sebelumnya.
<i>Actors:</i>	Anggota kelas
<i>Trigger:</i>	Anggota kelas memilih opsi untuk melihat leaderboard dari daftar tugas yang tersedia dalam kelas.
<i>Standard process:</i>	1. Anggota kelas membuka halaman daftar tugas dalam kelas tertentu. 2. Sistem menampilkan daftar tugas yang tersedia untuk kelas tersebut. 3. Anggota kelas memilih tugas yang diinginkan untuk melihat leaderboardnya. 4. Sistem mengambil informasi leaderboard tugas dari database. 5. Sistem menampilkan leaderboard tugas, termasuk peringkat, nama siswa, dan skor. 6. Anggota kelas dapat melihat peringkat mereka dan rekan-rekan mereka berdasarkan skor tugas.
<i>Alternative processes:</i>	1. Sistem menampilkan pesan kesalahan bahwa tugas atau leaderboard tidak ditemukan. 2. Siswa kembali ke halaman daftar tugas dan dapat memilih tugas lain.

**Tabel 4.8** Deskripsi Use Case Melihat Riwayat Submission

<i>Name:</i>	Melihat Riwayat Submission
<i>Short Description:</i>	Siswa dapat melihat riwayat pengumpulan tugas mereka, termasuk detail seperti tanggal pengumpulan, status penilaian, dan nilai.
<i>Precondition:</i>	1. Siswa telah terdaftar dan masuk ke dalam sistem. 2. Siswa sudah terdaftar di kelas yang terkait dengan tugas tersebut. 3. Siswa telah mengumpulkan satu atau lebih tugas dalam sistem.
<i>Postcondition:</i>	Berhasil: Siswa berhasil melihat riwayat <i>submission</i> untuk tugas yang diinginkan. Gagal: Siswa tetap berada di halaman sebelumnya dengan pesan kesalahan yang sesuai.
<i>Error situations:</i>	1. Tugas atau riwayat <i>submission</i> tidak ditemukan atau tidak tersedia. 2. Terjadi masalah pada sistem atau server saat mengambil informasi riwayat <i>submission</i> .
<i>System state in the event of an error:</i>	Sistem menampilkan pesan kesalahan yang sesuai di halaman sebelumnya.
<i>Actors:</i>	Siswa
<i>Trigger:</i>	Siswa memilih opsi untuk melihat riwayat submission dari daftar tugas yang tersedia dalam kelas.
<i>Standard process:</i>	1. Siswa membuka halaman daftar tugas dalam kelas tertentu. 2. Sistem menampilkan daftar tugas yang tersedia untuk kelas tersebut. 3. Siswa memilih tugas yang diinginkan untuk melihat riwayat <i>submission</i> nya. 4. Sistem mengambil informasi riwayat <i>submission</i> tugas dari database. 5. Sistem menampilkan riwayat <i>submission</i> , termasuk tanggal pengumpulan, status penilaian, dan nilai. 6. Siswa dapat melihat semua detail terkait riwayat <i>submission</i> untuk tugas tersebut.
<i>Alternative processes:</i>	1. Sistem menampilkan pesan kesalahan bahwa tugas atau riwayat <i>submission</i> tidak ditemukan. 2. Siswa kembali ke halaman daftar tugas dan dapat memilih tugas lain.

**Tabel 4.9** Deskripsi Use Case Melihat Anggota Kelas

<i>Name:</i>	Melihat Anggota Kelas
<i>Short Description:</i>	Anggota kelas dapat melihat daftar anggota yang terdaftar dalam suatu kelas, termasuk informasi dasar seperti nama, peran, dan status kehadiran.
<i>Precondition:</i>	1. Pengguna telah terdaftar dan masuk ke dalam sistem. 2. Pengguna sudah terdaftar di kelas yang ingin dilihat anggotanya.
<i>Postcondition:</i>	Berhasil: Pengguna berhasil melihat daftar anggota kelas yang diinginkan. Gagal: Pengguna tetap berada di halaman sebelumnya dengan pesan kesalahan yang sesuai.
<i>Error situations:</i>	Kelas tidak ditemukan atau tidak tersedia.
<i>System state in the event of an error:</i>	Sistem menampilkan pesan kesalahan yang sesuai di halaman sebelumnya.
<i>Actors:</i>	Anggota Kelas
<i>Trigger:</i>	Pengguna memilih opsi untuk melihat anggota kelas dari halaman kelas.
<i>Standard process:</i>	1. Pengguna membuka halaman kelas tertentu dari <i>dashboard</i> atau menu utama. 2. Sistem menampilkan opsi untuk melihat anggota kelas. 3. Pengguna memilih opsi "Lihat Anggota Kelas". 4. Sistem mengambil informasi daftar anggota kelas dari database. 5. Sistem menampilkan daftar anggota kelas, termasuk nama, peran, dan foto. 6. Pengguna dapat melihat semua informasi terkait anggota kelas tersebut.
<i>Alternative processes:</i>	1. Sistem menampilkan pesan kesalahan bahwa kelas tidak ditemukan. 2. Pengguna kembali ke halaman utama atau dashboard dan dapat memilih kelas lain.

**Tabel 4.10** Deskripsi Use Case Membuat Submission

<i>Name:</i>	Membuat Submission
<i>Short Description:</i>	Siswa dapat mengumpulkan tugas yang telah dikerjakan dengan mengunggah file program <i>java</i> .
<i>Precondition:</i>	1. Siswa telah terdaftar dan masuk ke dalam sistem. 2. Pengguna sudah terdaftar di kelas yang terkait dengan tugas tersebut.
<i>Postcondition:</i>	Berhasil: <i>Submission</i> berhasil dibuat dan disimpan di sistem, dan siswa menerima konfirmasi pengumpulan. Gagal: <i>Submission</i> tidak dibuat, dan siswa tetap berada di halaman <i>submission</i> dengan pesan kesalahan yang sesuai.
<i>Error situations:</i>	File yang diunggah tidak memenuhi kriteria (misalnya, ukuran file terlalu besar atau format file tidak didukung).
<i>System state in the event of an error:</i>	Sistem menampilkan pesan kesalahan yang sesuai di halaman <i>submission</i> .
<i>Actors:</i>	Siswa
<i>Trigger:</i>	Siswa memilih opsi untuk mengumpulkan tugas dari halaman tugas yang tersedia dalam kelas.
<i>Standard process:</i>	1. Siswa membuka halaman tugas dalam kelas tertentu. 2. Sistem menampilkan detail tugas dan opsi untuk membuat <i>submission</i> . 3. Siswa memilih opsi "Buat Submission". 4. Sistem menampilkan form <i>submission</i> yang memungkinkan siswa mengunggah file jawaban. 5. Siswa mengunggah <i>file</i> , lalu menekan tombol "Kirim". 6. Sistem memverifikasi bahwa <i>file</i> yang diunggah memenuhi kriteria yang ditentukan. 7. Jika valid, sistem menyimpan <i>submission</i> ke <i>database</i> . 8. Sistem menampilkan pesan konfirmasi bahwa <i>submission</i> berhasil dikumpulkan. 9. Siswa dapat melihat <i>submission</i> yang telah dikumpulkan di halaman riwayat <i>submission</i> .
<i>Alternative processes:</i>	1. Sistem menampilkan pesan kesalahan bahwa tugas tidak ditemukan atau tidak tersedia. 2. Siswa kembali ke halaman daftar tugas dan dapat memilih tugas lain.

**Tabel 4.11** Deskripsi *Use Case* Membuat Tugas

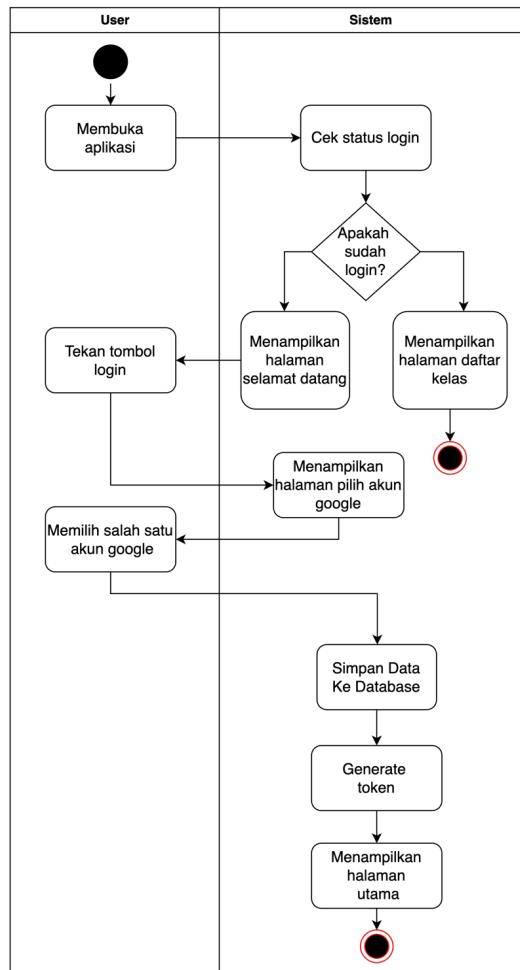
<i>Name:</i>	Membuat Tugas
<i>Short Description:</i>	Pengajar dapat membuat tugas baru dalam sistem, yang akan diberikan kepada siswa dalam kelas tertentu.
<i>Precondition:</i>	1. Pengajar telah terdaftar dan masuk ke dalam sistem dengan hak akses yang sesuai. 2. Pengajar sudah terdaftar di kelas di mana tugas akan dibuat.
<i>Postcondition:</i>	Berhasil: Tugas baru berhasil dibuat dan ditampilkan dalam daftar tugas di kelas yang sesuai. Gagal: Tugas tidak dibuat, dan pengajar tetap berada di halaman pembuatan tugas dengan pesan kesalahan yang sesuai.
<i>Error situations:</i>	Pengajar tidak mengisi semua <i>input</i> yang diperlukan.
<i>System state in the event of an error:</i>	Sistem menampilkan pesan kesalahan yang sesuai di halaman pembuatan tugas.
<i>Actors:</i>	Pengajar dan Pemilik Kelas
<i>Trigger:</i>	Pengajar memilih opsi "Buat Tugas Baru" dari halaman detail kelas.
<i>Standard process:</i>	1. Pengajar membuka halaman pembuatan tugas dengan memilih opsi "Buat Tugas Baru". 2. Sistem menampilkan form pembuatan tugas dengan berbagai input yang diperlukan (misalnya, judul tugas, deskripsi, tanggal mulai, tanggal jatuh tempo, dan <i>template testcase</i> ). 3. Pengajar mengisi semua bidang yang diperlukan pada form. 4. Pengajar menekan tombol "Simpan" atau "Buat Tugas". 5. Sistem memverifikasi bahwa semua bidang yang diperlukan telah diisi dengan data yang valid. 6. Sistem menyimpan data tugas baru ke dalam database. 7. Tugas baru muncul dalam daftar tugas yang tersedia di sistem untuk kelas tersebut.
<i>Alternative processes:</i>	1. Sistem menampilkan pesan kesalahan yang menunjukkan bahwa ada <i>inputan</i> yang belum diisi. 2. Pengajar melengkapi <i>inputan</i> yang diperlukan dan melanjutkan dari langkah 4 Proses Standar.

#### **4.2.7 Activity Diagram**

*Activity diagram* menggambarkan alur yang terjadi pada sebuah proses sistem dengan menampilkan serangkaian aktivitas, tindakan, keputusan, dan kontrol aliran yang terjadi pada aplikasi. *Activity diagram* dirancang berdasarkan interaksi-interaksi yang terjadi antara pengguna dengan aplikasi, beberapa interaksi tersebut:

1. *Activity diagram* autentikasi

*Activity diagram* autentikasi merupakan alur proses bagaimana cara pengguna masuk ke dalam aplikasi melalui sebuah akun *Google*. Terdapat dua kondisi, ketika pengguna sudah *login* sebelumnya maka akan langsung menampilkan halaman daftar kelas, tetapi ketika belum ada *session login* maka akan diminta *login* terlebih dahulu. *Activity diagram* autentikasi dapat dilihat pada Gambar 4.6.

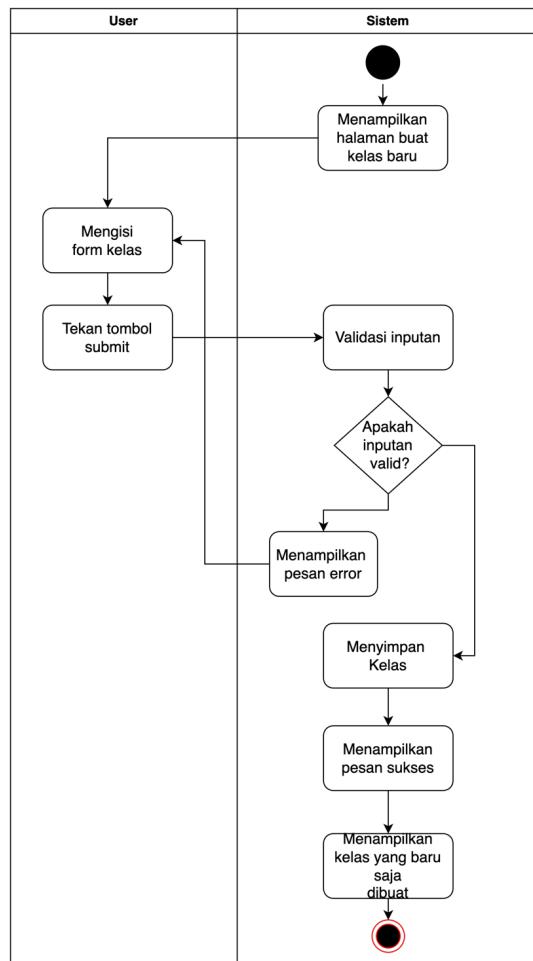


**Gambar 4.6 Activity Diagram Autentikasi**

## 2. Acitivity diagram membuat kelas

*Activity diagram* membuat kelas merupakan alur proses untuk membuat kelas, pengguna cukup mengisi *form* berupa nama kelas dan deskripsi, ketika inputan

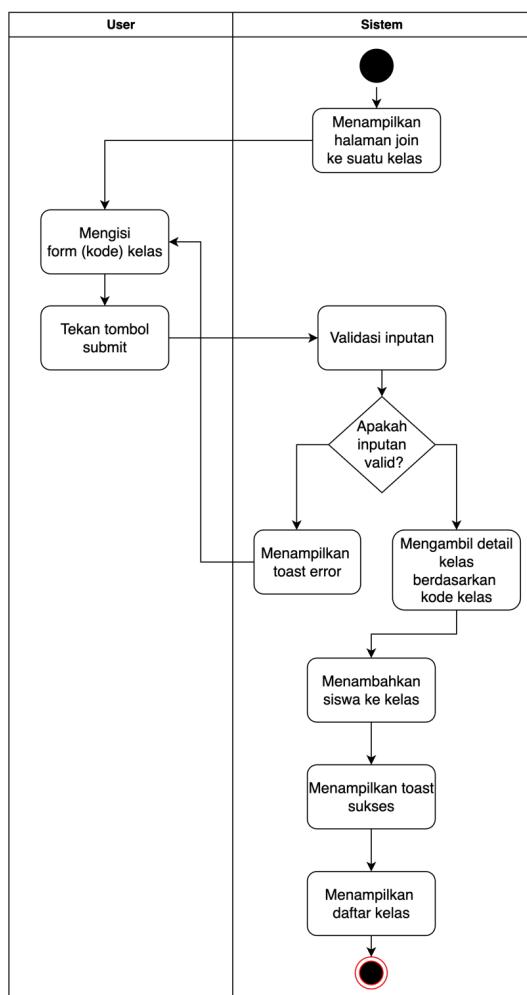
valid maka sistem akan membuatkan kelas untuk pengguna tersebut. *Activity diagram* membuat kelas dapat dilihat pada Gambar 4.7.



Gambar 4.7 *Activity Diagram* Membuat Kelas

3. *Activity diagram* bergabung menggunakan kode kelas

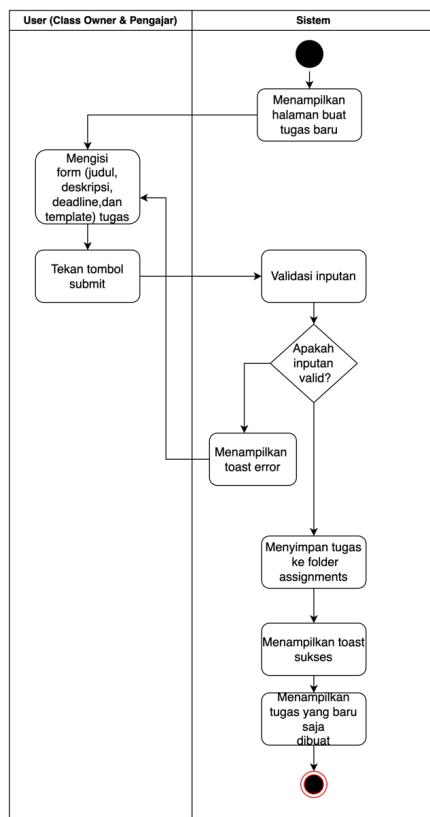
*Activity diagram* bergabung menggunakan kode kelas merupakan alur proses untuk pengguna lebih mudah bergabung ke suatu kelas hanya dengan memasukkan kode kelas. Ketika kode kelas valid maka pengguna akan didaftarkan sebagai siswa pada kelas tersebut. *Activity diagram* bergabung menggunakan kode kelas dapat dilihat pada Gambar 4.8.



**Gambar 4.8 Activity Diagram Bergabung Menggunakan Kode Kelas**

4. *Activity diagram* membuat tugas

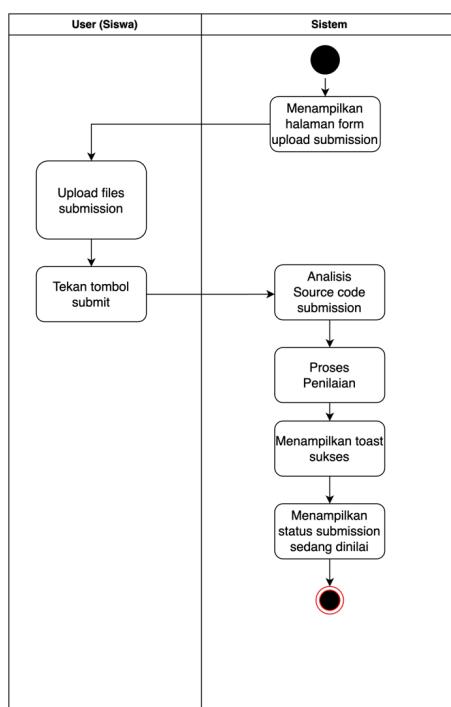
*Activity diagram* membuat kelas merupakan alur proses pengajar dan pemilik kelas untuk membuat tugas baru pada suatu kelas. Pengajar atau guru akan memasukkan detail dari tugas beserta *template testcase* tugas, jika *form* isian valid maka akan menyimpan tugas tersebut ke *database*. *Activity diagram* membuat tugas dapat dilihat pada Gambar 4.9.



**Gambar 4.9 Activity Diagram Membuat Tugas**

##### 5. Activity diagram membuat submission

Activity diagram membuat *submission* merupakan alur proses siswa untuk membuat *submission* atau pengumpulan tugas pada suatu tugas. Siswa akan mengirimkan *file* kode program *Java* dan sistem akan memasukkan kedalam *queue* atau antrian untuk dinilai secara otomatis dan menyimpan hasil penilaian ke *database*. Activity diagram membuat *submission* dapat dilihat pada Gambar 4.10.



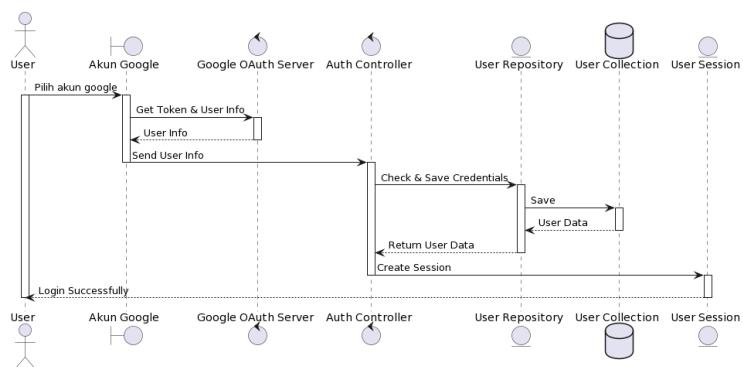
Gambar 4.10 Activity Diagram Membuat Submission

#### 4.2.8 Sequence Diagram

*Sequence diagram* menggambarkan interaksi antar objek-objek dalam aplikasi dalam pesan-pesan yang dikirimkan antar objek serta urutan waktu penyampaian pesan yang dapat diketahui. Berikut interaksi yang digambarkan menggunakan *sequence diagram*:

##### 1. *Sequence diagram* autentikasi

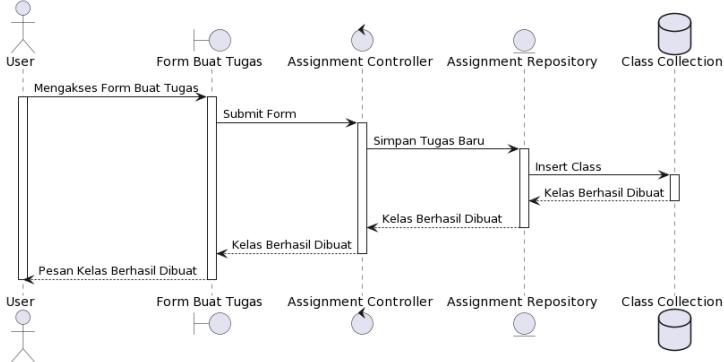
*Sequence diagram* autentikasi menggambarkan interaksi dan komponen yang terlibat untuk proses autentikasi. *Sequence diagram* autentikasi dapat dilihat pada Gambar 4.11.



Gambar 4.11 *Sequence Diagram* Autentikasi

##### 2. *Sequence diagram* membuat kelas

*Sequence diagram* membuat kelas merupakan interaksi antar komponen ketika pembuatan sebuah kelas. *Sequence diagram* membuat kelas dapat dilihat pada Gambar 4.12.



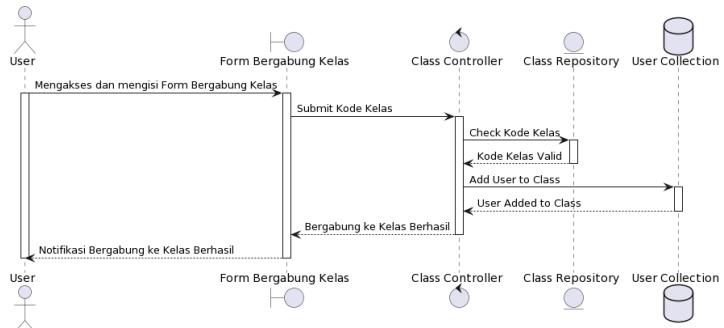
**Gambar 4.12 Sequence Diagram Membuat Kelas**

### 3. Sequence diagram bergabung menggunakan kode kelas

Sequence diagram bergabung menggunakan kode kelas merupakan interaksi antar komponen ketika seorang *user* ingin bergabung ke sebuah kelas.

Sequence diagram bergabung menggunakan kode kelas dapat dilihat pada

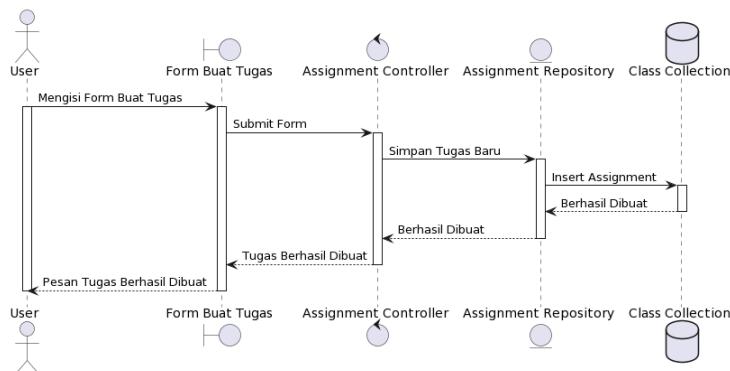
Gambar 4.13.



**Gambar 4.13 Sequence Diagram Bergabung Menggunakan Kode Kelas**

### 4. Sequence diagram membuat tugas

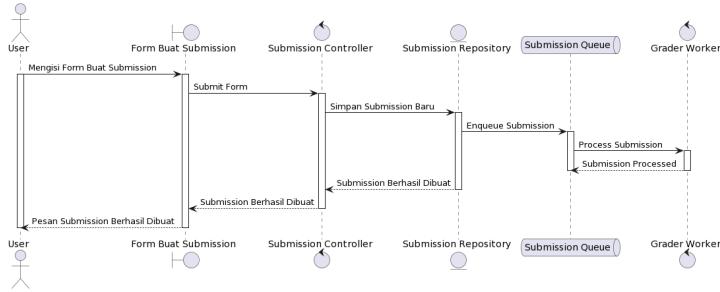
*Sequence diagram* membuat tugas merupakan interaksi antar komponen ketika pembuatan sebuah tugas atau *assignment*. *Sequence diagram* membuat tugas dapat dilihat pada Gambar 4.14.



**Gambar 4.14 Sequence Diagram Membuat Tugas**

##### 5. *Sequence diagram* membuat *submission*

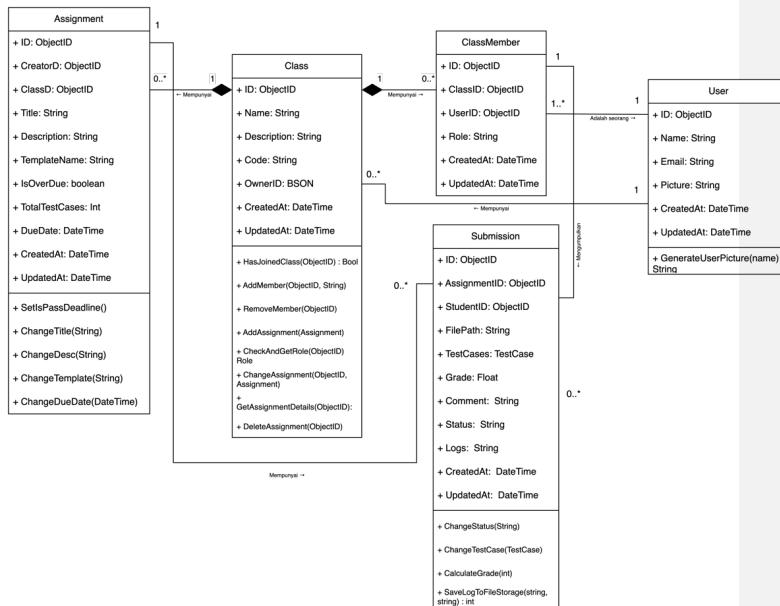
*Sequence diagram* membuat *submission* merupakan interaksi antar komponen ketika pembuatan atau pengiriman sebuah submission. Terdapat komponen *queue* tempat antrian untuk *submission* dan *worker process* untuk memproses *submission* dari *queue*. Untuk *Sequence diagram* membuat *submission* dapat dilihat pada Gambar 4.15.



**Gambar 4.15 Sequence Diagram Membuat Submission**

#### 4.2.9 Class Diagram

*Class diagram* menggambarkan struktur aplikasi dengan menuliskan kelas sistem, atribut yang dimiliki, metode serta hubungan antar objek yang terjadi pada aplikasi. *Class diagram* aplikasi *auto grader* yang dapat dilihat pada Gambar 4.16.



**Gambar 4.16 Class Diagram**

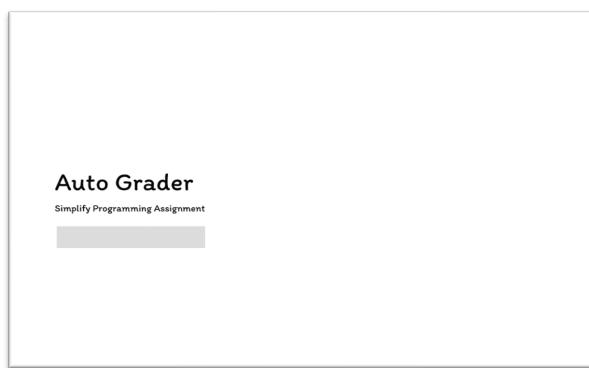
#### **4.2.10 User Interface Design**

*User interface design* merupakan bagian aplikasi yang berinteraksi langsung dengan pengguna. Tujuannya untuk memberikan gambaran tampilan aplikasi yang akan dibangun dan memberikan kemudahan pada proses implementasi aplikasi dengan cara mengubah hasil desain ke dalam bentuk kode aplikasi. Perancangan *user interface* aplikasi *auto grader* menggunakan *wireframe*.

*Wireframe* merupakan struktur dasar website atau disebut juga *blueprint*. Hasil dari tahapan *user interface design* adalah sebagai berikut:

1. *Autentication Page*

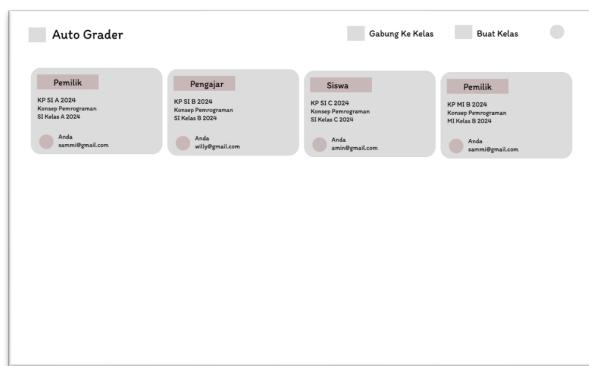
Halaman autentikasi terdiri dari komponen teks untuk menampilkan nama aplikasi, deskripsi, dan tombol untuk login yang ketika di klik secara otomatis akan menggunakan akun *Google* yang dipilih untuk masuk atau daftar ke aplikasi. *User interface design* halaman autentikasi dapat dilihat pada Gambar 4.17.



**Gambar 4.17** *Wireframe* Halaman Autentikasi

## 2. Class page

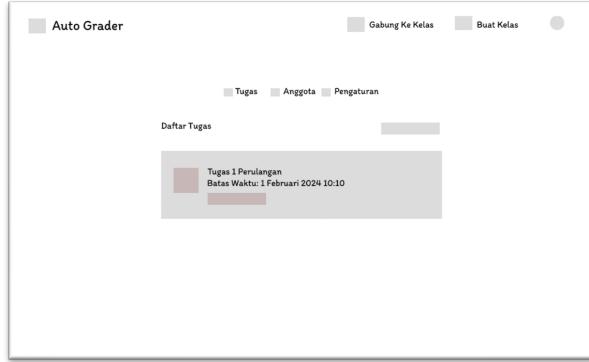
Halaman kelas terdiri dari kumpulan komponen *card* yang didalamnya terdapat informasi kelas seperti status pengguna, nama kelas, deskripsi kelas, dan informasi *owner* atau pemilik dari kelas tersebut. Daftar kelas yang ditampilkan adalah kelas yang pernah dibuat atau kelas yang pengguna telah bergabung di dalamnya. *User interface design* halaman kelas dapat dilihat pada Gambar 4.18.



Gambar 4.18 Wireframe Halaman Kelas

## 3. Assignment Page

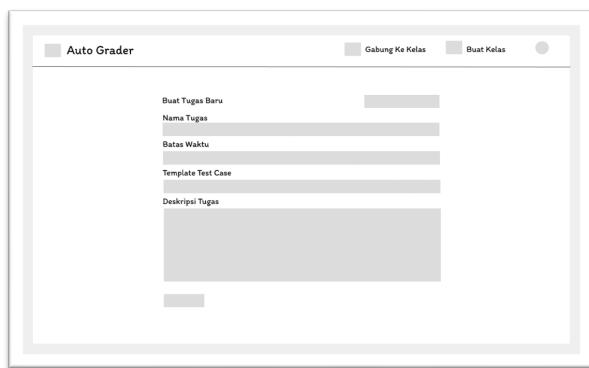
Halaman tugas terdiri dari kumpulan komponen *card* tugas yang telah dibuat oleh pengajar dari kelas tersebut. Halaman penugasan berisi informasi nama tugas, batas waktu, tombol buat tugas untuk navigasi ke halaman pembuatan tugas dan tombol *leaderboard* untuk navigasi ke halaman peringkat siswa dari tugas tersebut. *User interface design* halaman tugas dapat dilihat pada Gambar 4.19.



Gambar 4.19 Wireframe Halaman Tugas

#### 4. Create Assignment page

Halaman pembuatan tugas terdapat komponen *form* yang perlu diisi untuk membuat tugas baru. Form tersebut mencakup pengisian informasi untuk nama tugas, deskripsi tugas, *template* tugas, waktu pengumpulan dan tombol untuk mengirim form tersebut untuk disimpan. *User interface design* halaman pembuatan tugas dapat dilihat pada Gambar 4.20.

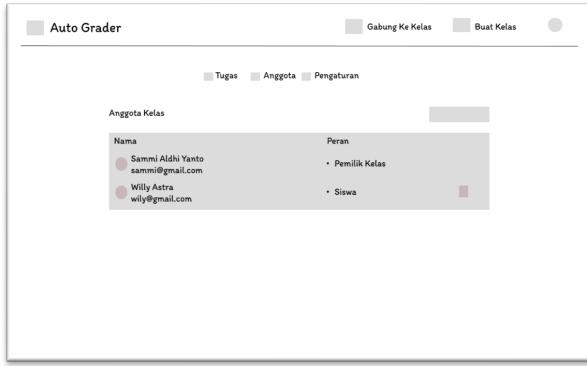


Gambar 4.20 Wireframe Halaman Buat Tugas

#### 5. Class member page

Commented [Y1]: Ini pindah ke halaman berikutnya

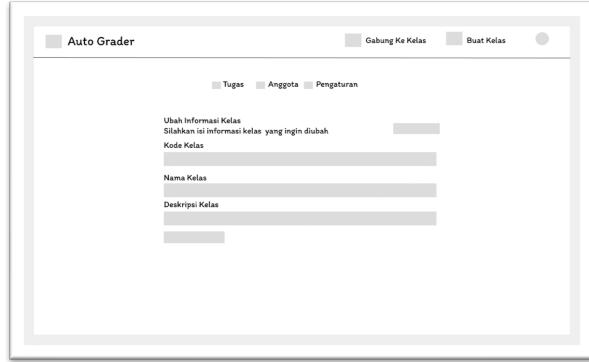
Halaman anggota kelas terdapat komponen *list* yang berisi daftar anggota kelas yang telah tergabung yang menampilkan nama, email, dan peran dari anggota kelas tersebut. Khusus untuk *owner* atau pemilik kelas maka akan ada tombol tambah dan hapus anggota kelas. *User interface design* halaman anggota kelas dapat dilihat pada Gambar 4.21.



**Gambar 4.21** Wireframe Halaman Anggota Kelas

#### 6. *Setting page*

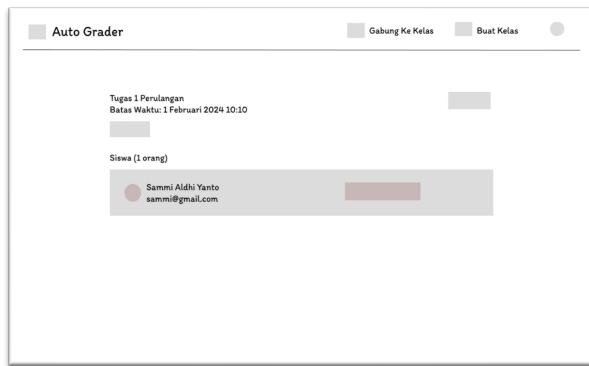
Halaman pengaturan terdapat form yang berisi informasi kelas seperti nama dan deskripsi kelas yang hanya bisa diubah oleh *owner* dari kelas tersebut. *User interface design* halaman pengaturan kelas dapat dilihat pada Gambar 4.22.



Gambar 4.22 Wireframe Halaman Pengaturan Kelas

#### 7. Student submission page

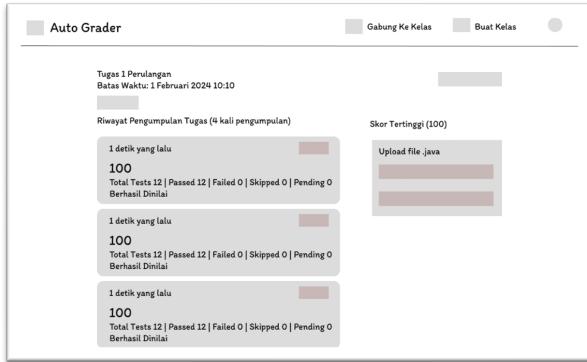
Halaman pengumpulan terdapat komponen daftar siswa pada suatu kelas beserta yang berisi nama dan email siswa tersebut beserta tombol untuk melihat riwayat dari pengumpulan yang pernah dilakukan. *User interface design* halaman pengumpulan dapat dilihat pada Gambar 4.23.



Gambar 4.23 Wireframe Halaman Daftar Submission Siswa

#### 8. Submission history page

Halaman riwayat pengumpulan terdapat daftar komponen *card* riwayat pengumpulan siswa yang berisi informasi seperti nilai dari pengumpulan, informasi total *testcase*, *log* pemrosesan, dan *form* untuk mengunggah *submission* baru. *User interface design* halaman riwayat pengumpulan kelas dapat dilihat pada Gambar 4.24.



Gambar 4.24 Wireframe Halaman Daftar Riwayat Pengumpulan Tugas

#### 9. Leaderboard page

Halaman peringkat terdapat daftar komponen *card* yang berisi daftar peringkat siswa dari suatu tugas yang telah diurutkan berdasarkan nilai dan waktu pengumpulan. *User interface design* halaman peringkat dapat dilihat pada Gambar 4.25.



**Gambar 4.25** Wireframe Halaman Papan Peringkat

### 4.3 Implementasi

Tahapan ini merupakan implementasi dari fase *tactical design* dalam *domain driven design*. *Tactical design* merujuk pada aspek-aspek, alat, dan teknik yang membantu menerjemahkan pemahaman tentang *domain* ke dalam perangkat lunak.

#### 4.3.1 Entities

*Entities* adalah objek yang mempunyai identitas. Kesamaan di antara entities ditentukan oleh identitasnya. *User* adalah salah satu kata benda yang ditemukan pada aplikasi *auto grader*. Setiap *user* dalam aplikasi memiliki identitas yang unik dan *email* digunakan sebagai elemen identifikasi yang tetap pada setiap proses dalam *domain* tersebut. Oleh karena itu dapat disimpulkan bahwa *user* memenuhi kriteria sebagai suatu *entities*. *Entities* bersifat *mutable*, yang berarti atribut-atributnya dapat berubah seiring waktu. Berikut adalah daftar entitas pada aplikasi *auto grader*:

1. *User*

Commented [Y2]: Ini juga, pinch ke halaman berikutnya

Entitas *User* memiliki atribut *email* menjadi identitas unik yang digunakan untuk membedakan setiap pengguna. Berikut struktur entitas *User* yang ditulis menggunakan bahasa pemrograman *Go*.

```
type User struct {
    ID      ID   `bson:"_id"`
    Name    string `bson:"name"`
    Email   string `bson:"email"`
    Picture string `bson:"picture"`
    CreatedAt time.Time `bson:"created_at"`
    UpdatedAt time.Time `bson:"updated_at"`
}
```

## 2. Assignment

Entitas *Assignment* memiliki atribut ID menjadi identitas unik yang digunakan untuk membedakan setiap tugas. Berikut struktur entitas *Assignment* yang ditulis menggunakan bahasa pemrograman *Go*.

```
type Assignment struct {
    ID      ID   `bson:"_id"`
    CreatorID ID   `bson:"creator_id"`
    Title    string `bson:"title"`
    Description string `bson:"description"`
    TemplateName string `bson:"template_name"`
    IsOverDue  bool  `bson:"_"`
    TotalTestCases int64 `bson:"total_test_cases"`
    DueDate    time.Time `bson:"due_date"`
    CreatedAt   time.Time `bson:"created_at"`
    UpdatedAt   time.Time `bson:"updated_at"`
}
```

## 3. Submission

Entitas *Submission* memiliki atribut ID menjadi identitas unik yang digunakan untuk membedakan setiap pengumpulan tugas. Berikut struktur entitas *Submission* yang ditulis menggunakan bahasa pemrograman *Go*.

```

type Submission struct {
    ID      ID `bson:"_id"`
    AssignmentID ID `bson:"assignment_id"`
    StudentID   ID `bson:"student_id"`

    FilePath string      `bson:"file_path"`
    TestCases TestCase     `bson:"test_cases"`
    Grade    float64     `bson:"grade"`
    Comment   string      `bson:"comment"`
    Status    SubmissionStatus `bson:"status"`
    Logs      string      `bson:"logs"`
    CreatedAt time.Time   `bson:"created_at"`
    UpdatedAt time.Time   `bson:"updated_at"`
}

```

#### 4.3.2 Value Objects

*Value objects* adalah objek yang menyimpan kumpulan nilai. Kesamaan antar *value objects* adalah kesamaan dari semua nilainya. *Value object* tidak mempunyai *identifier* dan bersifat *immutable*. Dalam aplikasi *auto grader* hanya ditemukan satu *value object* yaitu *Test Case*. Berikut Struktur *Test Case*.

```

type TestCase struct {
    Passed  int64 `bson:"passed"`
    Failures int64 `bson:"failures"`
    Errors  int64 `bson:"errors"`
    Skipped  int64 `bson:"skipped"`
}

```

#### 4.3.3 Aggregates

Agregat merujuk pada sekelompok objek *domain* yang dapat diperlakukan sebagai satu entitas untuk beberapa perilaku tertentu. Agregat bertindak sebagai batas transaksi untuk objek-objek *domain* di dalamnya. Pada aplikasi *auto grader* hanya ditemukan satu agregat yaitu *Class*. Berikut struktur *Class*.

```

type Class struct {
    ID      `bson:"_id"`
    Name    string `bson:"name"`
    Description string `bson:"description"`
    Code    string `bson:"code"`

    Owner   ID      `bson:"owner"`
    Teachers []ID    `bson:"teachers"`
    Students []ID    `bson:"students"`
    Assignments []*Assignment `bson:"assignments"`

    CreatedAt time.Time `bson:"created_at"`
    UpdatedAt time.Time `bson:"updated_at"`
}

```

#### 4.3.4 Factories

*Factories* merupakan mekanisme untuk mengonstruksi atau membangun *instance* dari *objek domain*. *Factory* dapat diimplementasikan melalui pembuatan konstruktor. Pada aplikasi *auto grader* terdapat 2 *factories* utama yaitu *factory* untuk membuat objek *class* dan *assignment*.

```

func NewClass(
    name,
    description,
    code string,
    owner primitive.ObjectID,
) (*Class, error) {

    return &Class{
        ID:      primitive.NewObjectID(),
        Name:    name,
        Description: description,
        Code:    code,

        Owner:   owner,
        Teachers: make([]primitive.ObjectID, 0),
        Students: make([]primitive.ObjectID, 0),
        Assignments: make([]*Assignment, 0),

        CreatedAt: helper.NewTime(),
        UpdatedAt: helper.NewTime(),
    }, nil
}

```

#### 4.3.5 Repositories

*Repositories* adalah bagian dari kode yang mengandung logika yang diperlukan untuk mengakses berbagai sumber data. Berikut merupakan kontrak dari implementasi *class repository*.

```
type IClassRepository interface {
    Save(
        ctx context.Context,
        class *domain.Class,
    ) error

    GetById(
        ctx context.Context,
        classID primitive.ObjectID,
    ) (*domain.Class, error)

    GetByCode(
        ctx context.Context,
        code string,
    ) (*domain.Class, error)

    GetAllClassesByUser(
        ctx context.Context,
        userID primitive.ObjectID,
    ) ([]*domain.Class, error)

    DeleteByID(
        ctx context.Context,
        classID primitive.ObjectID,
    ) error

    GetAssignmentLeaderboard(
        ctx context.Context,
        assignmentID primitive.ObjectID,
    ) ([]*model.GetAssignmentLeaderboardReadModel, error)

    GetGradingSummary(
        ctx context.Context,
        assignmentID primitive.ObjectID,
        students []primitive.ObjectID,
    ) ([]*model.GetGradingSummaryReadModel, error)

    GetAssignment(
        ctx context.Context,
        classID, assignmentID primitive.ObjectID,
    ) (*domain.Assignment, error)
}
```

#### 4.3.6 Services

*Service* merupakan *layer* atau komponen yang digunakan untuk menyediakan fungsionalitas atau *business logic*. Berikut merupakan kontrak dari *class service*.

```
type IClassService interface {
    GetAllClassesByUser(
        ctx context.Context,
        userID primitive.ObjectID,
    ) ([]*model.GetClassesReadModel, error)

    GetClassByID(
        ctx context.Context,
        classID primitive.ObjectID,
    ) (*domain.Class, error)

    GetClassDetailsReadModel(
        ctx context.Context,
        classID, userID primitive.ObjectID,
    ) (*model.GetClassDetailsReadModel, error)

    CreateNewClass(
        ctx context.Context,
        OwnerID primitive.ObjectID,
        req *model.CreateNewClassDTO,
    ) error

    DeleteClass(
        ctx context.Context,
        classID primitive.ObjectID,
    ) error

    UpdateClassDetails(
        ctx context.Context,
        ClassID primitive.ObjectID,
        req *model.UpdateClassDTO,
    ) error

    AddMember(
        ctx context.Context,
        req *model.AddMemberClassDTO,
    ) error

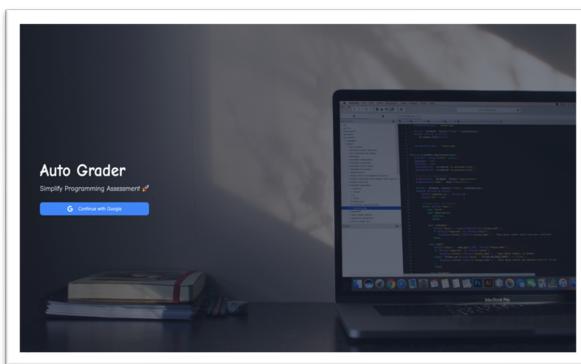
    RemoveMember(
        ctx context.Context,
        classID, userID primitive.ObjectID,
    ) error
}
```

#### **4.3.7 User Interface**

*User interface* merupakan implementasi dari *user interface design* yang telah dibuat pada tahap perancangan *user interface design*.

##### **1. Autentication Page**

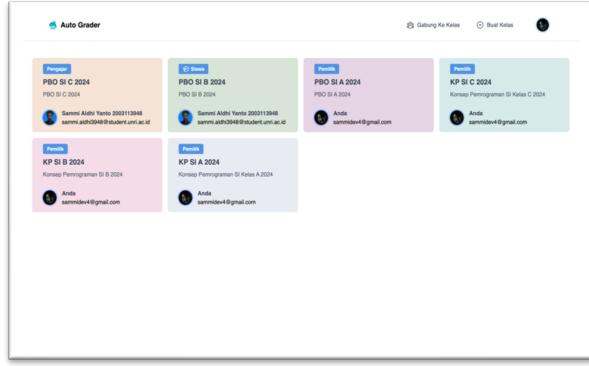
Halaman autentikasi adalah halaman *default* ketika pengguna belum masuk ke dalam aplikasi. Pada halaman autentikasi terdapat tombol “*Continue with Google*” yang secara otomatis akan menggunakan akun *Google* yang dipilih untuk masuk atau daftar ke aplikasi. Tampilan halaman autentikasi dapat dilihat pada Gambar 4.26.



**Gambar 4.26** Halaman Autentikasi

##### **2. Class page**

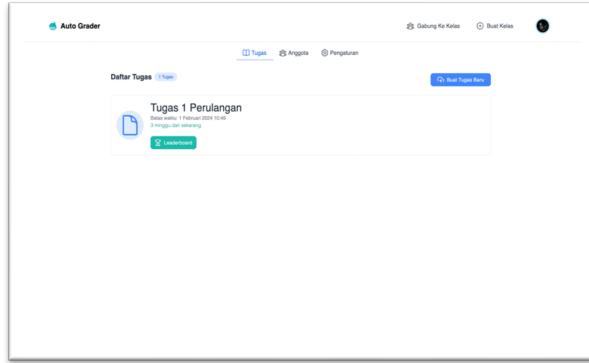
Halaman kelas menampilkan daftar kelas yang pernah dibuat atau kelas yang pengguna telah bergabung di dalamnya. Masing-masing kelas menampilkan informasi seperti nama, status pengguna, dan pemilik kelas tersebut. Tampilan halaman kelas dapat dilihat pada Gambar 4.27.



**Gambar 4.27** Halaman Kelas

### 3. Assignment Page

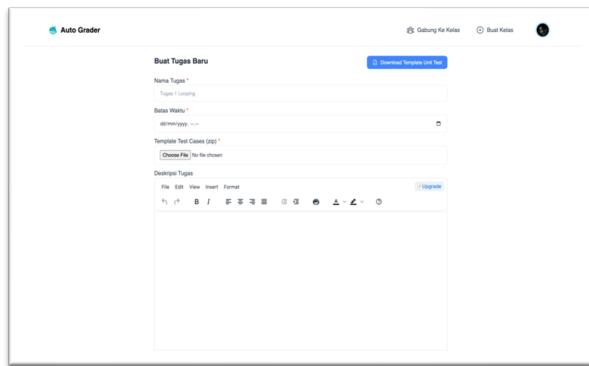
Halaman tugas menampilkan daftar tugas yang telah dibuat oleh pengajar dari kelas tersebut. Setiap daftar tugas menampilkan informasi nama tugas, batas waktu pengumpulan, tombol *leaderboard* dan tombol untuk membuat tugas baru. Tampilan halaman tugas dapat dilihat pada Gambar 4.28.



**Gambar 4.28** Halaman Daftar Tugas

#### 4. Create Assignment page

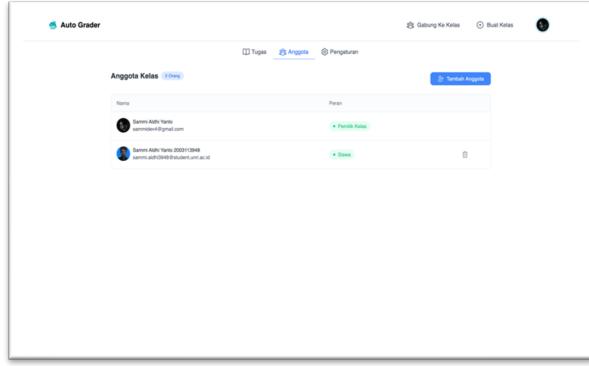
Halaman buat tugas menampilkan *form* yang perlu diisi untuk membuat tugas baru. *Form* tersebut mencakup informasi seperti nama tugas, deskripsi, *template testcase*, dan batas waktu pengumpulan. Tampilan halaman buat tugas dapat dilihat pada Gambar 4.29.



Gambar 4.29 Halaman Buat Tugas

#### 5. Class member page

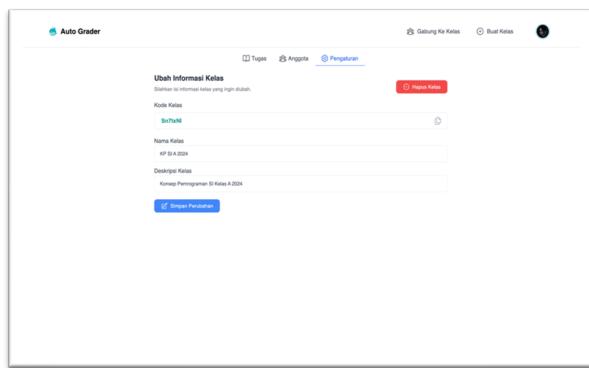
Halaman anggota kelas menampilkan daftar anggota kelas yang telah tergabung. Khusus untuk *owner* kelas akan muncul tambah dan hapus anggota kelas. Tampilan halaman anggota kelas dapat dilihat pada Gambar 4.30.



**Gambar 4.30** Halaman Daftar Anggota Kelas

#### 6. *Setting page*

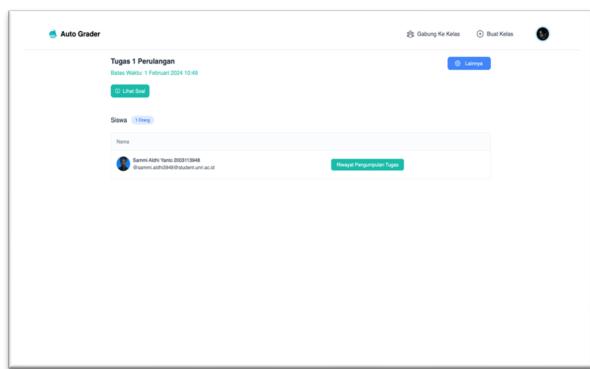
Halaman pengaturan menampilkan informasi kelas seperti nama dan deskripsi kelas yang hanya bisa diubah oleh pemilik dari kelas tersebut. Tampilan halaman pengaturan kelas dapat dilihat pada Gambar 4.31.



**Gambar 4.31** Halaman Pengaturan Kelas

#### 7. Student submission page

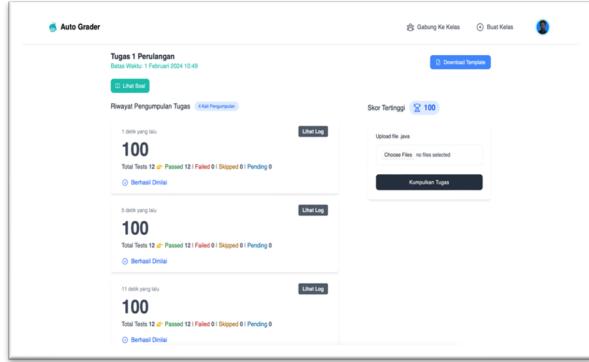
Halaman submisi menampilkan daftar siswa pada suatu kelas beserta tombol untuk melihat riwayat dari submission yang pernah dilakukan. Tampilan halaman submisi dapat dilihat pada Gambar 4.32.



**Gambar 4.32** Halaman Submission Mahasiswa

#### 8. Submission history page

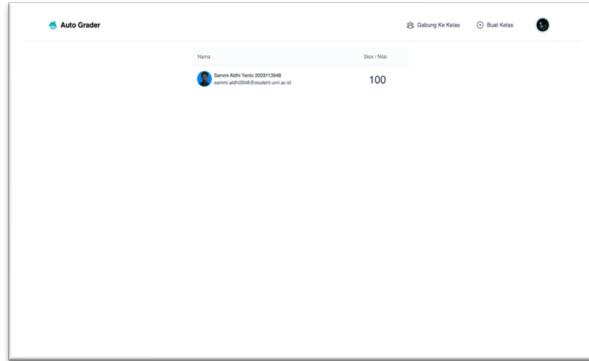
Halaman riwayat submisi menampilkan riwayat pengiriman jawaban tugas yang berisi informasi seperti nilai, total  *testcase* ,  *log*  pemrosesan, beserta  *form*  untuk mengunggah  *submission*  baru. Tampilan halaman riwayat submisi dapat dilihat pada Gambar 4.33.



Gambar 4.33 Halaman Riwayat Submission

#### 9. Leaderboard page

Halaman peringkat menampilkan urutan peringkat siswa dari suatu tugas yang telah diurutkan berdasarkan peringkat dan waktu pengumpulan. Tampilan halaman peringkat dapat dilihat pada Gambar 4.34.



Gambar 4.34 Halaman Peringkat

## 4.4 Pengujian

Pengujian merupakan tahapan evaluasi yang bertujuan untuk menilai kualitas, keandalan, dan fungsionalitas aplikasi. Tujuan utama dari pengujian ini ialah untuk menemukan kesalahan dalam aplikasi sehingga dapat diperbaiki sebelum diimplementasikan kepada pengguna. Metode pengujian aplikasi yang digunakan adalah *black box testing* dan *user acceptance testing*.

### 4.4.1 Black Box Testing

#### 1. Fitur Autentikasi

Terdapat fungsionalitas fitur autentikasi yang telah diuji, hasil pengujian dapat dilihat pada Tabel 4.12.

**Tabel 4.12** Hasil *Testing* Fitur Autentikasi

No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
1.	Menekan tombol <i>Continue With Google</i>	Menampilkan pilihan akun <i>google</i> yang teraut pada perangkat dan <i>redirect</i> ke halaman daftar kelas	Sesuai	<i>Valid</i>
2.	Menekan tombol keluar pada halaman <i>home</i>	Diarahkan ke halaman <i>login</i>	Sesuai	<i>Valid</i>

#### 2. Fitur Kelas

Terdapat fungsionalitas fitur kelas yang telah diuji, hasil pengujian dapat dilihat pada Tabel 4.13.

**Tabel 4.13** Hasil *Testing* Fitur Kelas

No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
1.	Menekan tombol buat kelas pada <i>navbar</i>	Menampilkan <i>pop up form</i> untuk membuat kelas	Sesuai	<i>Valid</i>
2.	Menekan tombol buat kelas pada <i>form</i>	Aplikasi berhasil menjalankan proses dan menampilkan kelas dibuat	Sesuai	<i>Valid</i>
3.	Tidak mengisi kode kelas pada formulir	Menampilkan pesan <i>error</i>	Sesuai	<i>Valid</i>
4.	Menekan tombol bergabung	Aplikasi berhasil menjalankan proses dan menampilkan kelas yang sesuai dengan kode kelas	Sesuai	<i>Valid</i>
5.	<i>Submit form</i> dengan kode kelas yang sudah pernah ditambahkan sebelumnya	Menampilkan <i>pop up error</i> dengan pesan “Anda sudah bergabung pada kelas ini”	Sesuai	<i>Valid</i>
6.	Menekan salah satu nama pada daftar kelas	Menampilkan daftar tugas pada kelas tersebut	Sesuai	<i>Valid</i>
7.	Menghapus nama kelas dan menekan tombol simpan perubahan	Menampilkan pesan <i>error</i> pada kolom nama	Sesuai	<i>Valid</i>

Lanjutan Tabel 4.13 Hasil *Testing* Fitur Kelas

No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
8.	Mengisi informasi kelas dengan lengkap dan menekan tombol simpan perubahan	Aplikasi berhasil menjalankan proses dan menampilkan informasi kelas yang terbaru	Sesuai	<i>Valid</i>
9.	Menekan tombol hapus kelas	Muncul <i>prompt</i> persetujuan	Sesuai	<i>Valid</i>

### 3. Fitur Anggota Kelas

Terdapat fungsionalitas fitur anggota kelas yang telah diuji, hasil pengujian dapat dilihat pada Tabel 4.14.

**Tabel 4.14** Hasil *Testing* Fitur Anggota Kelas

No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
1.	Menekan <i>tab</i> Anggota	Menampilkan daftar siswa, guru, dan <i>owner</i> kelas	Sesuai	<i>Valid</i>
2.	Tidak mengisi alamat <i>email</i> pada <i>form</i> tambah anggota	Menampilkan pesan <i>error</i> pada kolom <i>email</i>	Sesuai	<i>Valid</i>
3.	Menekan tombol “Tambahkan” dengan mengisi <i>email</i> yang belum terdaftar pada aplikasi	Menampilkan <i>pop up error</i> pengguna tidak ditemukan	Sesuai	<i>Valid</i>

Lanjutan Tabel 4.14 Hasil *Testing* Fitur Anggota Kelas

No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
4.	Menekan tombol “Tambahkan” dengan mengisi <i>email</i> yang terdaftar pada aplikasi tapi belum menjadi anggota kelas	Menampilkan <i>pop up</i> berhasil bergabung	Sesuai	<i>Valid</i>
5.	Menekan tombol “Tambahkan” dengan mengisi <i>email</i> yang terdaftar pada aplikasi dan telah menjadi anggota kelas	Menampilkan pesan <i>error</i> anda telah bergabung pada kelas ini	Sesuai	<i>Valid</i>
6.	Menekan tombol <i>trash</i> pada nama anggota kelas	Muncul <i>prompt</i> persetujuan	Sesuai	<i>Valid</i>
7.	Menekan tombol ok pada <i>prompt</i> persetujuan	Anggota kelas berhasil dihapus	Sesuai	<i>Valid</i>

#### 4. Fitur Tugas

Terdapat fungsionalitas fitur tugas yang telah diuji, hasil pengujian dapat dilihat pada Tabel 4.15.

**Tabel 4.15** Hasil *Testing* Fitur Tugas

No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
1.	Menekan tombol buat tugas baru	Menampilkan <i>form</i> untuk membuat tugas	Sesuai	<i>Valid</i>
2.	Mengosongkan <i>input</i> yang wajib	Menampilkan pesan <i>error</i> pada kolom	Sesuai	<i>Valid</i>

Lanjutan Tabel 4.15 Hasil Testing Fitur Tugas

No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
3.	Mengisi semua input yang wajib pada <i>form</i> dan menekan tombol Buat tugas	Tugas akan terbuat dan mengarahkan ke halaman daftar tugas	Sesuai	<i>Valid</i>
4.	Menekan tombol <i>leaderboard</i> pada suatu tugas	Menampilkan halaman <i>leaderboard</i>	Sesuai	<i>Valid</i>
5.	Menekan salah satu tugas (ketika pengguna merupakan Pengajar atau Pemilik Kelas)	Menampilkan daftar siswa dan tombol Riwayat pengumpulan tugas	Sesuai	<i>Valid</i>
6.	Menekan tombol riwayat pengumpulan tugas pada salah satu siswa (ketika pengguna merupakan Pengajar atau Pemilik Kelas)	Menampilkan halaman daftar riwayat pengumpulan beserta nilainya	Sesuai	<i>Valid</i>
7.	Menekan tombol log pada salah satu <i>submission</i> (ketika pengguna merupakan Pengajar atau Pemilik Kelas)	Menampilkan halaman detail dari pemrosesan tugas yang dilakukan oleh sistem	Sesuai	<i>Valid</i>
8.	Menekan salah satu tugas (ketika pengguna merupakan seorang siswa)	Menampilkan Riwayat pengumpulan dan <i>form</i> untuk mengumpulkan tugas	Sesuai	<i>Valid</i>

Lanjutan Tabel 4.15 Hasil Testing Fitur Tugas

No	Deskripsi Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
9.	Mengumpulkan Tugas ketika tugas belum melewati batas waktu <i>deadline</i> .	Sistem akan memproses pengumpulan	Sesuai	<i>Valid</i>
10.	Memilih <i>file</i> program <i>Java</i> dan menekan tombol "Kumpulkan Tugas" ketika tugas belum telah melewati batas waktu <i>deadline</i> .	Tombol tidak bisa di klik	Sesuai	<i>Valid</i>
11.	Menekan tombol unduh nilai	<i>File CSV</i> yang berisi nilai siswa akan di ter <i>download</i>	Sesuai	<i>Valid</i>
12.	Menekan tombol hapus tugas	Menampilkan <i>prompt</i> persetujuan untuk menghapus tugas	Sesuai	<i>Valid</i>

#### 4.4.2 User Acceptance Testing

Metode *user acceptance testing (UAT)* berguna untuk mengetahui tanggapan dari *user* terhadap sistem yang telah dibangun dengan cara menggunakan kuesioner. Pengujian *UAT* dilakukan dengan menyebarkan *form* kuesioner kepada 1 kelas pemrograman berorientasi objek yang berjumlah sebanyak 20 orang mahasiswa dan 1 orang dosen pengampu matakuliah tersebut. Kuesioner terdiri dari masing-masing 4 pertanyaan yang berkaitan dengan desain dan fitur aplikasi.

Daftar pertanyaan yang digunakan dalam kuisioner *user acceptance testing* dapat dilihat pada Tabel 4.16.

**Tabel 4.16** Pertanyaan Kuisioner *User Acceptance Testing*

Kode	Pertanyaan
<b>Desain</b>	
P1	Navigasi menu dalam aplikasi terstruktur dengan baik dan memudahkan dalam menemukan fitur yang dibutuhkan
P2	Desain warna yang digunakan sudah sesuai
P3	Tampilan aplikasi <i>auto grader</i> menarik
P4	Penggunaan tulisan ( <i>font</i> ) mudah dibaca
<b>Fitur</b>	
P5	Proses pengiriman dan penilaian tugas berjalan lancar
P6	Fitur-fitur dalam aplikasi dapat dioperasikan dengan cepat dan tanpa kesulitan
P7	Aplikasi dapat mengevaluasi kode program dengan cepat
P8	Proses login ke dalam aplikasi dilakukan dengan cepat dan mudah

Hasil UAT responden terhadap pertanyaan-pertanyaan yang diberikan dapat dilihat pada Tabel 4.17.

**Tabel 4.17** Hasil UAT Responden

Kode	Nilai Responden					Bobot
	STS x 1	TS x 2	C x 3	S x 4	SS x 5	
P1	0 x 1	0 x 2	5 x 3	13 x 4	3 x 5	82
P2	0 x 1	0 x 2	1 x 3	18 x 4	2 x 5	85
P3	0 x 1	0 x 2	6 x 3	12 x 4	3 x 5	81
P4	0 x 1	0 x 2	1 x 3	10 x 4	11 x 5	98
P5	0 x 1	3 x 2	7 x 3	11 x 4	1 x 5	76
P6	0 x 1	0 x 2	3 x 3	15 x 4	3 x 5	84
P7	0 x 1	1 x 2	9 x 3	9 x 4	2 x 5	75
P8	0 x 1	0 x 2	1 x 3	15 x 4	5 x 5	88

Pada Tabel 4.17 merupakan hasil UAT yang sudah dikalikan dengan bobot skala *Likert*. Data pada tabel ini kemudian digunakan untuk menghitung nilai

persentase berdasarkan rumus 2.1. Setelah mendapatkan nilai rata-rata, maka dibutuhkan perhitungan persentase menggunakan rumus pada 2.2, pertanyaan yang dilakukan untuk mendapatkan hasil kualitas sistem untuk layak untuk digunakan bagi pengguna. Adapun hasil persentase pertanyaan dapat dilihat pada Tabel 4.18.

**Tabel 4.18** Hasil Presentase Pertanyaan UAT

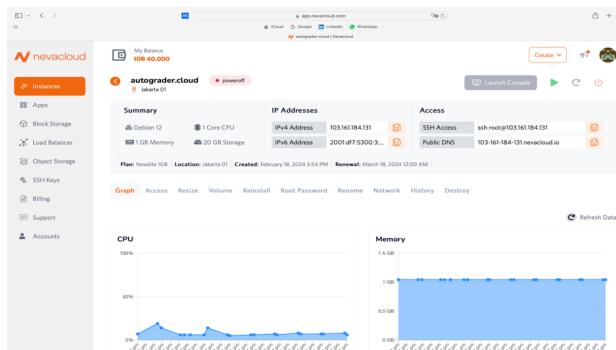
Kode	Nilai Rata - Rata	Persentase	Keterangan
P1	$82 / 21 = 3,90$	78%	Kategori Desain
P2	$85 / 21 = 4,05$	81%	Kategori Desain
P3	$81 / 21 = 3,86$	77%	Kategori Desain
P4	$98 / 21 = 4,67$	93%	Kategori Desain
P5	$76 / 21 = 3,62$	72%	Kategori Fitur
P6	$84 / 21 = 4,00$	80%	Kategori Fitur
P7	$75 / 21 = 3,57$	71%	Kategori Fitur
P8	$88 / 21 = 4,19$	84%	Kategori Fitur

Pada tabel 4.18 merupakan hasil persentase pertanyaan *user acceptance test*, dapat disimpulkan bahwa rata-rata persentase pertanyaan dengan kategori desain yaitu 82% dan rata-rata persentase pertanyaan dengan kategori fitur yaitu 77%, dan rata-rata total persentase pertanyaan dengan semua kategori yaitu 80%. Berdasarkan hasil dari rata-rata total persentase, maka aplikasi masuk dalam kategori “baik” berdasarkan kriteria interpretasi skor yang terdapat pada Tabel 2.6.

#### **4.5 Deployment**

Deployment merupakan proses memindahkan sebuah aplikasi atau sistem dari lingkungan pengembangan ke lingkungan produksi atau pengguna akhir. Proses ini melibatkan instalasi perangkat lunak, konfigurasi, dan penyesuaian agar aplikasi dapat berjalan dengan lancar. Proses *deployment* dimulai dengan *setup domain*, sistem operasi, dan spesifikasi *server* di *cloud provider*. Kemudian

instalasi perangkat lunak yang dibutuhkan seperti *Nginx*, *Apache Maven*, *JDK 11*, *PHP*, *Composer*, *Cert Bot*, *Golang*, *Redis*, *NodeJS*, dan *MongoDB*. Informasi spesifikasi *server* dari aplikasi *auto grader* dapat dilihat pada Gambar 4.35.



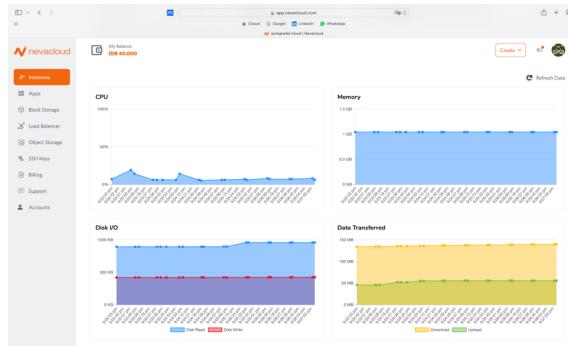
**Gambar 4.35** Spesifikasi Server

#### 4.6 Maintenance

*Maintenance* atau pemeliharaan perangkat lunak, adalah proses yang penting untuk memastikan bahwa aplikasi tetap beroperasi secara optimal dan dapat menyesuaikan diri dengan perubahan yang terjadi dalam lingkungan atau kebutuhan pengguna. Ini melibatkan pemantauan kinerja sistem secara berkala untuk mengidentifikasi dan mengatasi masalah yang mungkin terjadi, seperti *bug* atau kesalahan dalam program.

Selain itu, *maintenance* juga mencakup pelaksanaan pembaruan perangkat lunak secara teratur untuk memperbaiki keamanan dan kinerja serta menambahkan fitur baru sesuai dengan umpan balik dari pengguna. Dengan demikian, pemeliharaan perangkat lunak merupakan upaya berkelanjutan yang penting untuk memastikan aplikasi tetap relevan, aman, dan berfungsi sebagaimana mestinya.

sepanjang waktu. *Dashboard usage monitoring* yang disediakan oleh *Platform NevaCloud* dapat dilihat pada Gambar 4.36.



Gambar 4.36 Resource Monitoring

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Berdasarkan dari penelitian yang telah dilakukan serta hasil yang didapatkan, maka dapat diambil kesimpulan sebagai berikut:

1. Proses membangun aplikasi *auto grader* dilakukan dengan tahapan yang berurutan sebagai berikut: pengumpulan data dan analisis kebutuhan, perancangan dan design menggunakan *strategic design*, *ubiquitous language*, UML, dan *wireframe*, implementasi aplikasi menggunakan *tactical design*, pengujian menggunakan *black box testing* dan *user acceptance testing*, *deployment*, dan terakhir *maintenance*. berdasarkan tahapan yang dilakukan, maka diperoleh elemen yang dibutuhkan dalam perancangan dan pengembangan aplikasi *auto grader*. Pertama, Sub

Commented [Y3]: ini jawaban dari rumusan masalah #1

*Domain* dari aplikasi *auto grader* yang terdiri dari *user management*, *class management*, *assignment management*, *class member management*, *submission management*, dan *role permission management*. Kedua, Dari subdomain tersebut di *grouping* menjadi 3 *bounded context* yaitu *user context*, *class context*, dan *submission context*. Pembagian *bounded context* tersebut dilakukan berdasarkan kapabilitasnya. Daftar istilah pada *auto grader* yaitu *user*, *class*, *assignment*, *submission*, *owner*, *teacher*, *student*, *leaderboard*, *log*, *being graded*, *successfully graded*, dan *failed to grade*.

*Shared kernel* digunakan untuk komunikasi antar bounded context.

Commented [Y4]: Hapus saja

2. tambahkan penjelasan untuk menjawab rumusan masalah no #2:
3. Aplikasi *auto grader* telah menjalani serangkaian tes untuk mengevaluasi kualitas dan fungsionalitasnya. *Black box testing* dilakukan untuk menguji berbagai fitur dalam aplikasi, seperti fitur autentikasi, kelas, anggota kelas, dan tugas. Hasil pengujian dari masing-masing fitur telah direkam dalam tabel yang menunjukkan kesesuaian antara hasil yang diharapkan dengan hasil yang sebenarnya. Selanjutnya, *UAT* dilakukan untuk mengevaluasi respons dari pengguna terhadap aplikasi. Berdasarkan hasil persentase *UAT*, dapat disimpulkan bahwa rata-rata persentase pertanyaan dengan kategori desain adalah 82%, menunjukkan tingkat kepuasan yang baik terhadap aspek desain aplikasi. Sementara itu, rata-rata persentase pertanyaan dengan kategori fitur adalah 77%, mengindikasikan respons yang positif terhadap fitur-fitur yang disediakan. Total persentase pertanyaan dengan semua kategori adalah 80%. Berdasarkan hasil total persentase, aplikasi masuk dalam kategori “baik” berdasarkan kriteria interpretasi skor.

Commented [Y5]: Kesimpulan 3 bisa jadi hasil kesimpulan tambahan

## 5.2 Saran

Berdasarkan penelitian yang telah dilakukan, berikut beberapa saran yang dapat menjadi masukan untuk penelitian selanjutnya, adapun saran-saran tersebut yaitu:

1. Aplikasi dapat menerapkan gabungan antara metode *dynamic* dengan *static analysis* dalam proses *grading*. *Static analysis* tidak perlu mengkompilasi atau mengeksekusi kode.

Analisis ini hanya tentang melakukan analisis sintaksis dan semantik. Integrasi kedua metode ini akan memungkinkan evaluasi yang lebih holistik terhadap kualitas kode yang diajukan. Analisis statis akan membantu dalam mengidentifikasi potensi masalah sejak tahap awal pengembangan, sementara analisis dinamis akan memberikan pemahaman yang lebih mendalam tentang perilaku program saat berjalan. Kombinasi keduanya diharapkan dapat meningkatkan akurasi penilaian serta memberikan saran yang lebih relevan untuk perbaikan.

2. Menambahkan fitur deteksi plagiarisme dan umpan balik terhadap *submission* pengguna. Penambahan fitur ini akan membantu mencegah kecurangan akademik plagiarisme dengan mengidentifikasi kesamaan antara kode program dan juga memberikan kesempatan bagi pengguna untuk memperbaiki *submission* mereka terhadap masukan yang diberikan. Dengan demikian, integrasi fitur ini diharapkan dapat mendukung integritas dan etika akademik serta meningkatkan pembelajaran yang berkelanjutan dan berkualitas.

## DAFTAR PUSTAKA

- Adhiwibowo, W., & Daru, A. F. (2017). Metode Object Oriented Programming. In *JURNAL INFORMATIKA UPGRIS* (Vol. 3, Issue 2).
- Ahdan, S., Sucipto, A., & Agus Nurhuda, Y. (2019). *Game untuk Menstimulasi Kecerdasan Majemuk pada Anak (Multiple Intelligence) Berbasis Android Game to Stimulate Children's Multiple Intelligence Based on Android.*
- Aldriye, H., Alkhalfaf, A., & Alkhalfaf, M. (2019). Automated Grading Systems for Programming Assignments: A Literature Review. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 10, Issue 3). [www.ijacsa.thesai.org](http://www.ijacsa.thesai.org)
- Boyle Matthew. (2022). *Domain-Driven Design with Golang: Using Golang to Create Simple, Maintainable Systems to Solve Complex Business Problems.* Packt Publishing Limited.
- Caiza, J. C., & Alamo, J. M. Del. (2013). *Programming Assignments Automatic Grading: Review of Tools and Implementations.*
- Chandrasekaran, P., & Krishnan, K. (2022). *Domain-driven design with Java : a practitioner's guide : create simple, elegant, and valuable software solutions for complex business problems.*
- Endra, R. Y., Aprilinda, Y., Dharmawan, Y. Y., & Ramadhan, W. (2021). Analisis Perbandingan Bahasa Pemrograman PHP Laravel dengan PHP Native pada Pengembangan Website. *EXPERT: Jurnal Manajemen Sistem Informasi Dan Teknologi*, 11(1), 48. <https://doi.org/10.36448/expert.v11i1.2012>

- Feby Prasetya, A., & Lestari Dewi Putri, U. (2022). Perancangan Aplikasi Rental Mobil Menggunakan Diagram UML (Unified Modelling Language). In *DOI: ...* (Vol. 1, Issue 1).
- Juliawan Pawana, I. W. A., Wiharta, D. M., & Sastra, N. P. (2021). Identifikasi Kandidat Microservices Dengan Analisis Domain Driven Design. *Majalah Ilmiah Teknologi Elektro*, 20(2), 273. <https://doi.org/10.24843/mite.2021.v20i02.p11>
- Khononov, V., & Lerman, J. (2022). *Learning Domain-Driven Design Aligning Software Architecture and Business Strategy*.
- Michail, & Pervolarakis. (2023). *DEVELOPMENT OF AN AUTOMATIC CODE GRADING PLATFORM*.
- Nurjanah, S., & Zulfikli. (2017). Perencanaan Aplikasi Sistem Pendukung Keputusan untuk Menentukan Kriteria Pencairan Dana Kredit Nasabah BMT El-Ihsan. *OJS STMIK Pringsewu*, 5.
- Prayoga, H. W., & Akbar, R. J. (2021). Rancang Bangun Sistem MyITS Dorm Menggunakan Metode Domain Driven Design dan Onion Architecture. *JURNAL TEKNIK ITS*.
- Putri Yulandi, A. (n.d.). *Analisis Performa Backend Framework: Studi Komparasi Framework Golang dan Node.js*. 8, 155–168. <https://tunasbangsa.ac.id/ejurnal/index.php/jurasik>
- Rahmi, Z., Nukman, H., Teknik Informatika STMIK, P. U., Jl Alue Naga Desa Tibang, B., & Aceh, B. (2014). PERBANDINGAN METODOLOGY

KLASIK DAN AGILE DALAM PENGEMBANGAN SISTEM INFORMASI. In *Banda Aceh* (Vol. 24).

Saputra, D. R., Purnomo, W., Setiawan, N. Y., & Korespondensi, P. (2022). *Sistem Propagasi Anotasi pada Metadata Lineage untuk Manajemen Data Warehouse Annotation Propagation System on Lineage Metadata for Data Warehouse Management.* 9(7), 1741–1746.  
<https://doi.org/10.25126/jtiik.202296833>

Setiana, E., Rizki Ramadhan, M., & Yadi Rakhman, R. A. (2024). *Pengujian Perangkat Lunak Metode Black Box Pada Aplikasi Sistem Pakar Pola Latihan dan Asupan Makanan* (Vol. 18, Issue 1).

<https://journal.fkom.uniku.ac.id/ilkom>

Simangunsong, J., & Voutama, A. (2023). Rancang Bangun Sistem Informasi Online Marketplace Berbasis Web Application: Studi Kasus KLG Campus Residence. In *Jurnal Mahasiswa Teknik Informatika* (Vol. 7, Issue 2).

Wang, J., Zhao, Y., Tang, Z., & Xing, Z. (2020). Combining Dynamic and Static Analysis for Automated Grading SQL Statements. In *Taiwan Ubiquitous Information* (Vol. 5, Issue 4). <https://www.codeforces.com/>

Winata, E., & Setiawan, J. (2013). Analisis dan Perancangan Prototipe Aplikasi Tracking Bis Universitas Multimedia Nusantara pada Platform Android. *ULTIMA InfoSys*, IV(1).

Commented [Y6]: tolong di cek lagi sam, ini apakah harus 2 spasi semua atau 1 spasi dan 1.5 spasi untuk jeda antar satu rujukan dgn rujukan lainnya

## **LAMPIRAN**

**Lampiran 1.** Tabel Daftar Pertanyaan Kuesioner

Kode	Pertanyaan
<b>Desain</b>	
P1	Navigasi menu dalam aplikasi terstruktur dengan baik dan memudahkan dalam menemukan fitur yang dibutuhkan
P2	Desain warna yang digunakan sudah sesuai
P3	Tampilan aplikasi <i>auto grader</i> menarik
P4	Penggunaan tulisan ( <i>font</i> ) mudah dibaca
<b>Fitur</b>	
P5	Proses pengiriman dan penilaian tugas berjalan lancar
P6	Fitur-fitur dalam aplikasi dapat dioperasikan dengan cepat dan tanpa kesulitan
P7	Aplikasi dapat mengevaluasi kode program dengan cepat
P8	Proses login ke dalam aplikasi dilakukan dengan cepat dan mudah

**Lampiran 2.** Tabel Hasil Kuesioner

Alamat Email	P1	P2	P3	P4	P5	P6	P7	P8
Partisipan 1	4	4	3	4	4	4	4	4
Partisipan 2	4	4	3	4	4	4	3	4
Partisipan 3	4	4	4	4	2	4	4	4
Partisipan 4	4	4	4	4	3	3	3	4
Partisipan 5	5	5	4	5	4	5	4	5
Partisipan 6	3	4	4	4	4	4	3	4
Partisipan 7	4	3	3	4	3	4	4	3
Partisipan 8	5	4	5	5	3	4	5	4
Partisipan 9	4	4	3	5	4	4	4	4
Partisipan 10	4	4	3	4	3	4	3	4
Partisipan 11	4	4	4	5	5	4	4	5
Partisipan 12	4	4	4	4	3	4	3	4
Partisipan 13	4	4	4	4	4	4	4	4
Partisipan 14	5	5	5	5	4	5	4	4
Partisipan 15	3	4	4	5	3	4	3	5
Partisipan 16	4	4	4	5	4	4	3	4
Partisipan 17	4	4	4	5	4	4	4	5
Partisipan 18	4	4	3	5	4	4	3	5
Partisipan 19	4	4	3	5	4	4	4	4
Partisipan 20	3	4	4	4	3	3	3	4
Partisipan 21	3	4	4	4	4	5	5	4
Partisipan 22	3	4	4	4	2	4	3	4
Partisipan 23	4	4	3	4	2	3	2	4

**Lampiran 3.** Foto Dokumentasi Wawancara dan Demo Aplikasi Bersama *Domain Expert*.



