PROPOSAL KELOMPOK 3

TUGAS BESAR

Mata Kuliah Metode Formal

Oleh :

1302220101	Yoga Fikri Rizkulloh	SE-46-03
1302220113	Kyntar Barra Langit Lubis	SE-46-03
1302220152	Haidar Abdul Halim	SE-46-03
1302223119	M Faisal Shafwan Damanik	SE-46-03

PRODI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

A. Pendahuluan

Aplikasi ini dirancang untuk memudahkan pelacakan tiket untuk sebuah event yang dijual di dua situs web yang berbeda. Aplikasi ini akan membantu pengguna untuk memantau perbedaan harga tiket antara dua situs web tersebut dan memberikan notifikasi melalui email jika tiket di situs web pertama (Site A) lebih murah dibandingkan dengan tiket yang sama di situs web kedua (Site B). Dengan ini, pengguna dapat mengambil keputusan yang tepat saat membeli tiket untuk event yang sama.

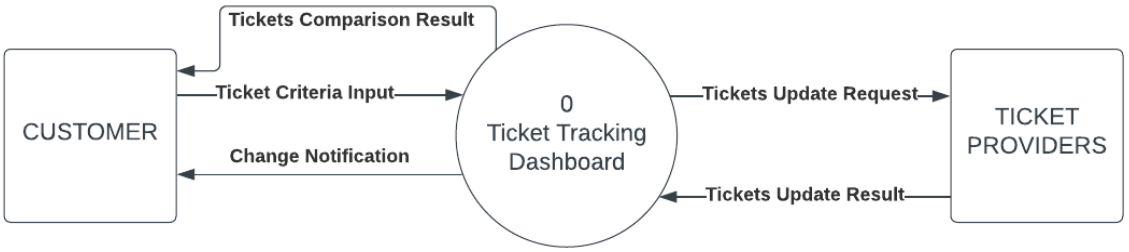
B. Goal

a. Memberikan kemudahan kepada pengguna untuk memantau harga tiket event yang sama di dua situs web berbeda.
b. Memberikan notifikasi kepada pengguna melalui email ketika tiket di Site A lebih murah daripada di Site B.
c. Menghemat waktu dan uang pengguna dengan memberikan informasi yang akurat tentang perbedaan harga tiket.
d. Membantu pengguna dalam pengambilan keputusan yang bijak saat membeli tiket untuk event yang diminati.
e. Memberikan pengalaman pengguna yang efisien dan efektif dalam proses pembelian tiket event.

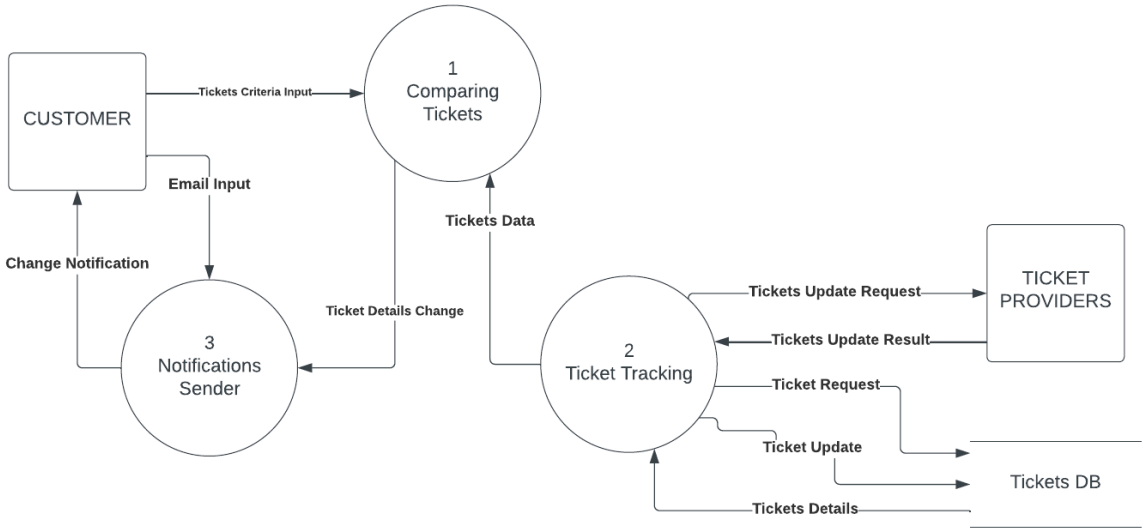
C. Pemodelan

a. Data Flow Diagram (DFD)

i. Level 0

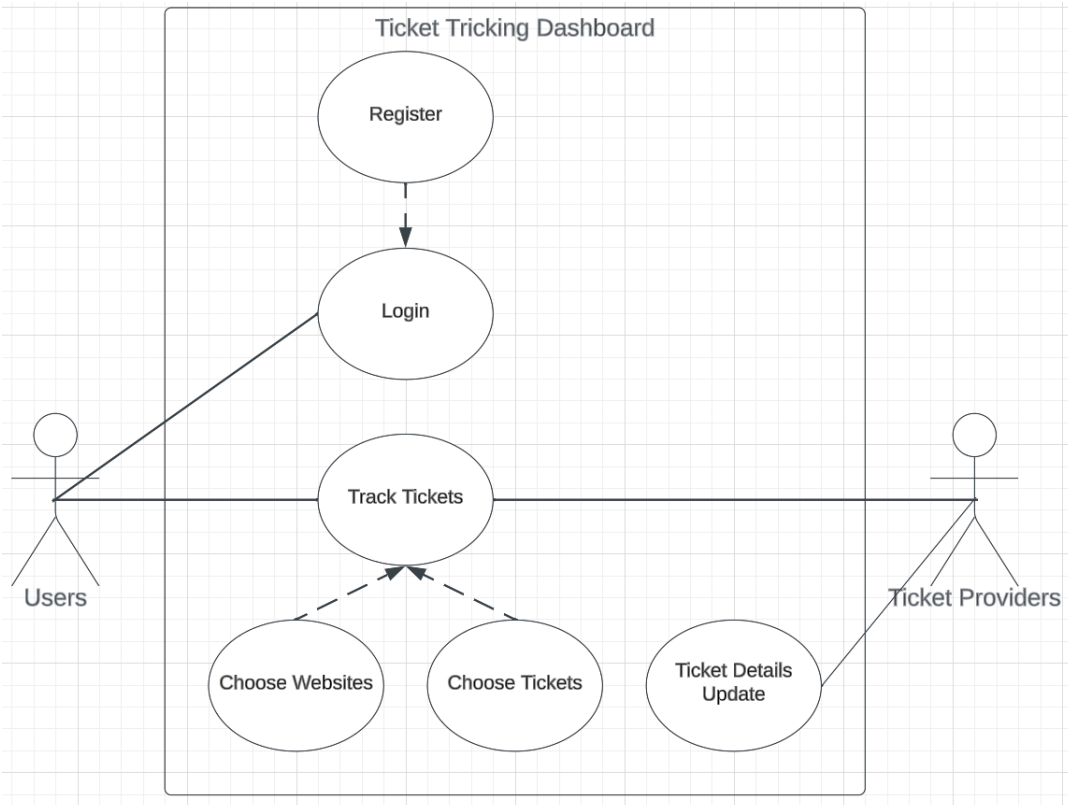


ii. Level 1

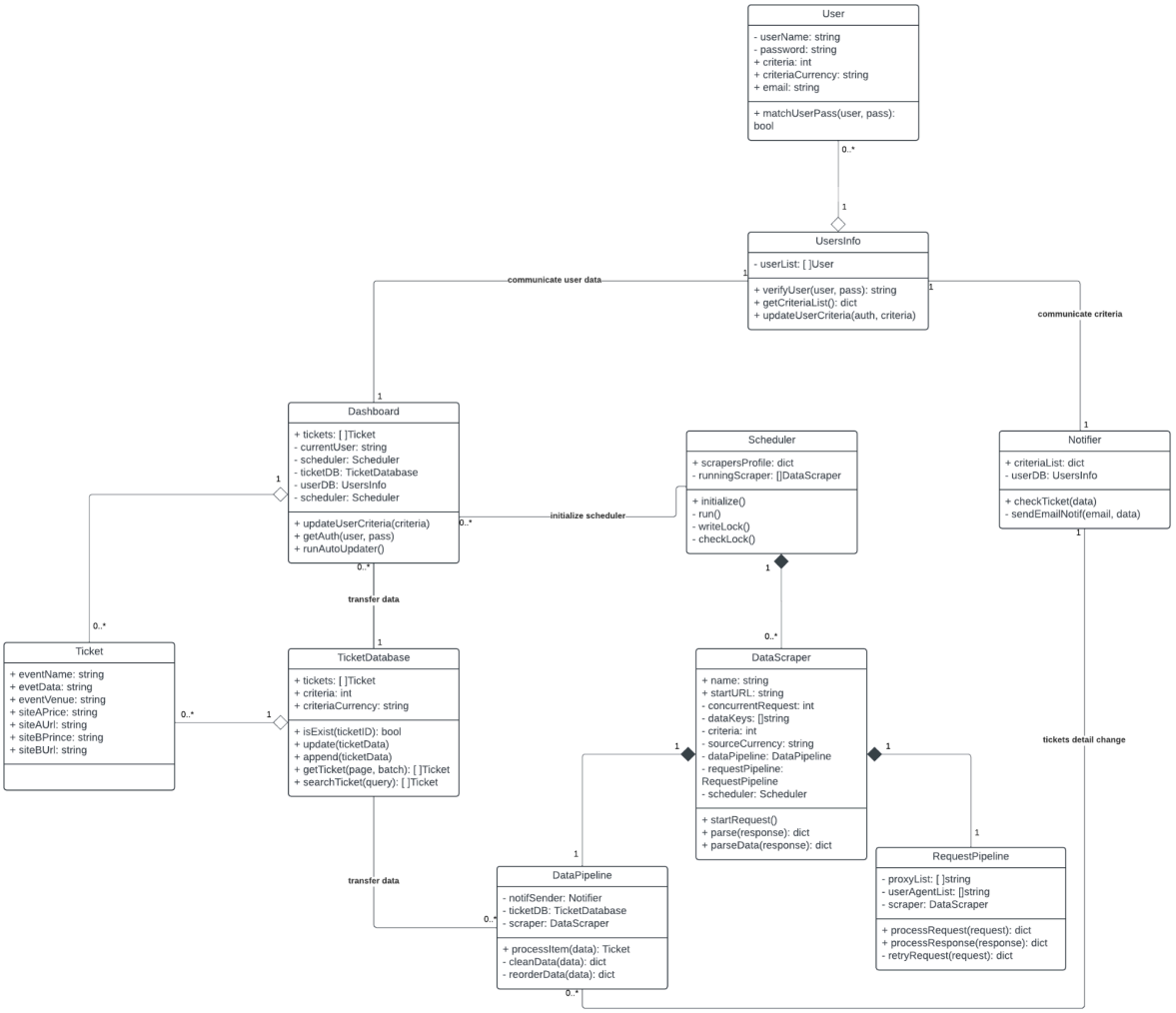


b. Unified Modelling Language

i. Use Case Diagram

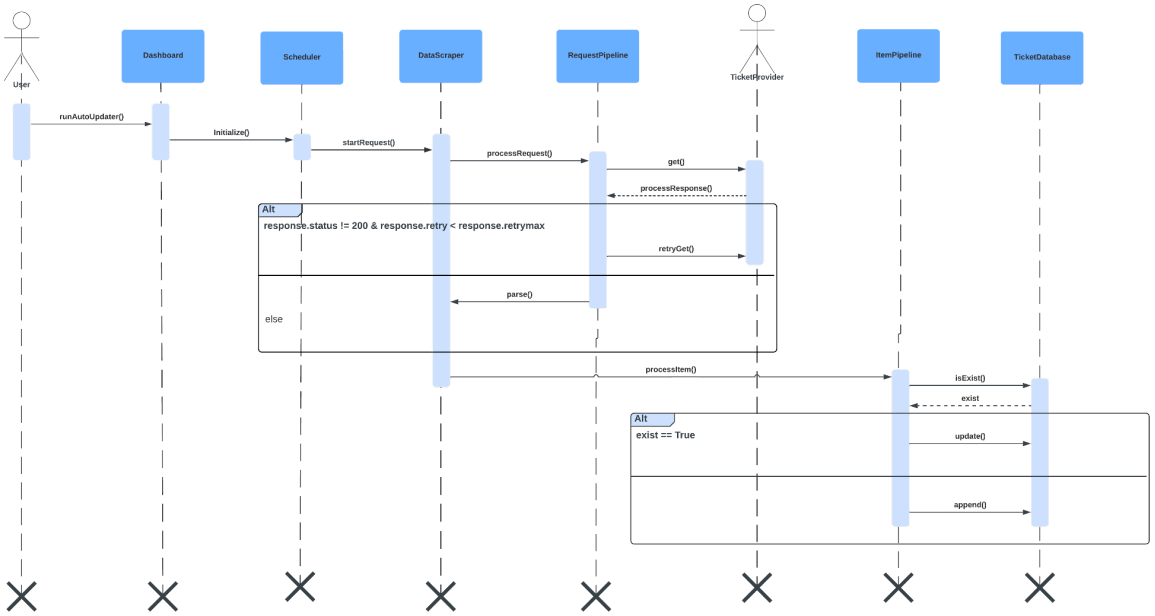


ii. Class Diagram

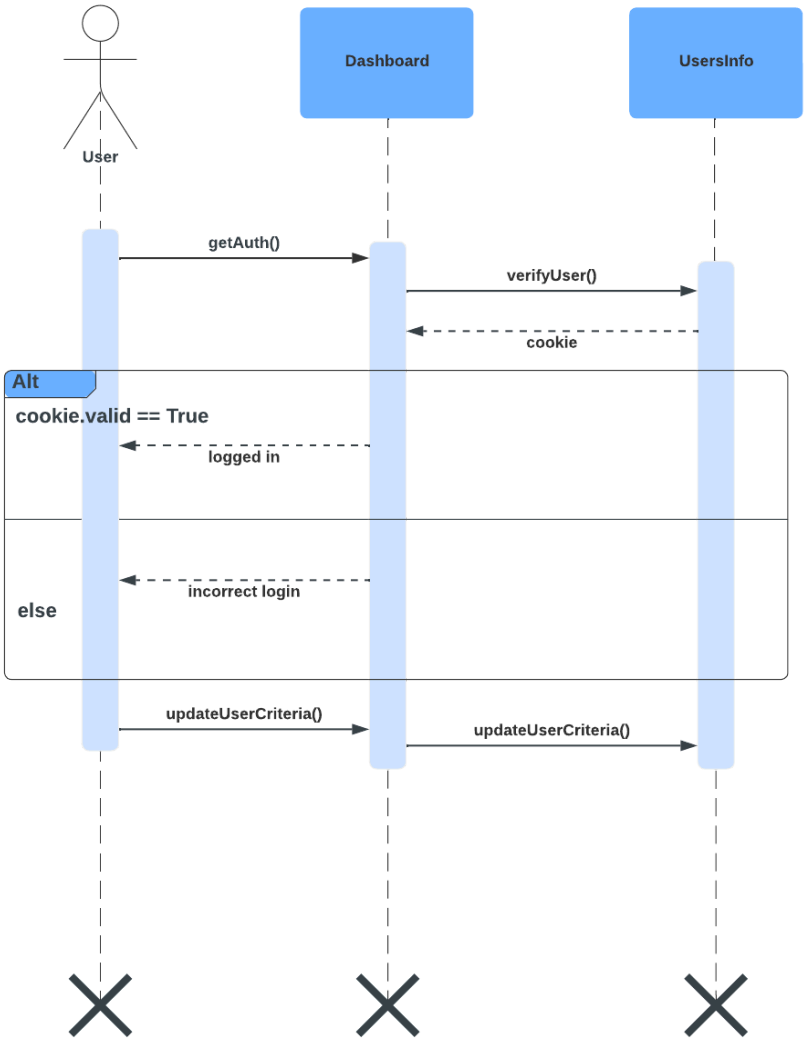


iii. Sequence Diagram

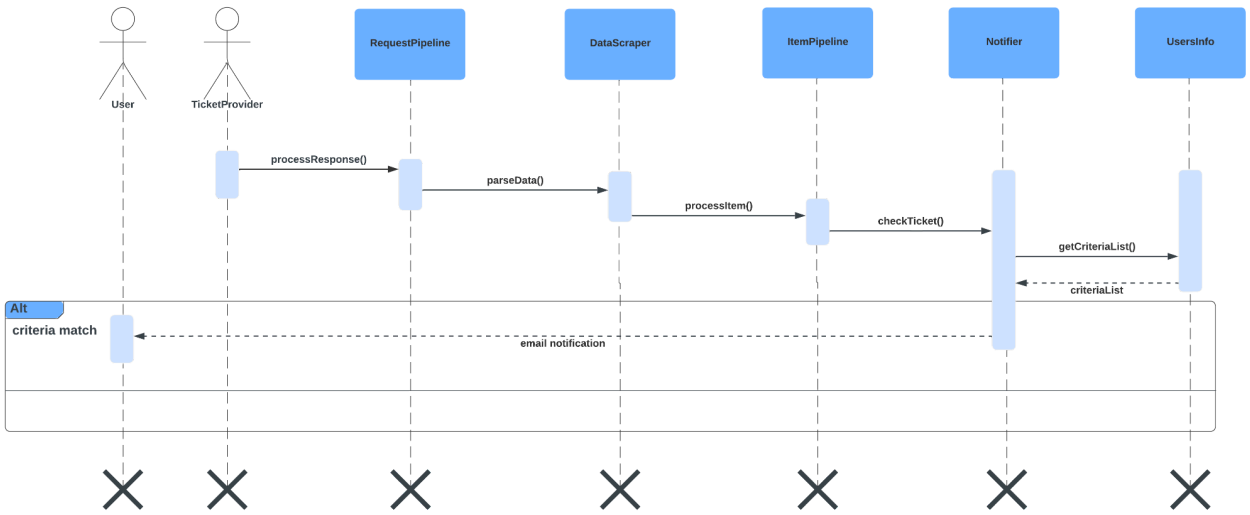
1. Data Scraping



2. Update Criteria



3. Notification



c. Alloy

```
one sig Text {}
one sig EncryptedText {}
one sig Dict {}

one sig UserInfo {
  usersList: some User
}

sig User {
  userId: Int,
  userName: Text,
  password: EncryptedText,
  criteria: Int,
  criteriaCurrency: Text,
  email: Text,
  notiFrom: some Ticket
} {
      userId > 0

      this in UserInfo.usersList
}

abstract sig NotificationSent {}
one sig Yes, No extends NotificationSent {}
sig Ticket {
 eventId: Int,
 price1: Int,
 price2: Int,
 notification: NotificationSent,
 source: one DataScraper
} {
// Memastikan kalau Ticket hanya akan muncul jika berada didalam
TicketDatabase.tickets
      this in TicketDatabase.tickets
// Memastikan kalau idnya tidak negatif
      eventId > 0
}
one sig Dashboard {
  tickets: some Ticket,
  currentUser: one User,
  scheduler: one Scheduler,
  userInfo: UserInfo
}
```

```
one sig TicketDatabase{
  tickets: some Ticket
}
one sig Scheduler{
      scrapersProfile: Dict
}
one sig Notifier {
      criteriaList: Dict,
      userDB: some User
}
sig DataScraper{
      name: Text,
      startURL: Text,
      concurrentRequest: Int,
   belong_to: set Scheduler,
   dataPipeline: one DataPipeline,
   requestPipeline: RequestPipeline
} {
// Memastikan kalau Datascraper hanya akan muncul jika berada
didalam Ticket.source
      this in Ticket.source
}
sig DataPipeline {
      ticketDB: TicketDatabase
} {
      this in DataScraper.dataPipeline
}
sig RequestPipeline {
      proxyList: Text,
      userAgentList: Text
} {
// Memastikan kalau RequestPipeline hanya akan muncul jika berada
didalam DataScraper.requestPipeline
      this in DataScraper.requestPipeline
}
fact TicketConstraint {
  all e: Ticket |
   e.price1 > 0 and e.price2 > 0
}
fact TicketSourceConstraint {
  all t: Ticket | some s: DataScraper | t.source = s
}
fact DSContraint {
 all d: DataScraper | some s: Scheduler | d.belong_to = s
}
fact Notification {
  all t: Ticket |
    (t.notification = Yes) =>
```

```
      some u: User | some n: u.notiFrom | t = n
}
fact ValidNotif {
  all u: User |
     all t: u.notiFrom | sub[t.price1, t.price2] >= u.criteria
}
fact NotifYes {
   all t: Ticket | all u: User | (t in u.notiFrom) =>
t.notification = Yes
}
pred CriteriaOneOrMore {
 all u: User |
   u.criteria > 5
}
pred CriteriaZero {
 all u: User |
   u.criteria <1
}
assert NoNotificationHighCriteria {
all u: User | u.criteria > 5 implies
// no t: Ticket |
//      (t.notification = Yes) =>
//          all u: User |
//               (t in u.notiFrom) =>
//                   (t.price1 - t.price2) >= u.criteria
no t: Ticket | t.notification = Yes
}
assert NotificationLowCriteria {
  all u: User | u.criteria < 0 implies
// some t: Ticket |
//      (t.notification = Yes) =>
//          all u: User |
//               (t in u.notiFrom) =>
//                   (t.price1 - t.price2) >= u.criteria
some t: Ticket | t.notification = Yes
}
// Memastikan tidak ada yang mempunyai set yang sama
fact noDuplicate {
     all t1, t2: Ticket | t1 != t2 =>
          t1.source not in t2.source &&
          t1.eventId not in t2.eventId
     all us1, us2: User | us1 != us2 =>
          us1.userId not in us2.userId
     all rp1, rp2: DataScraper | rp1 != rp2 =>
          rp1.requestPipeline not in rp2.requestPipeline
}

run CriteriaOneOrMore for 5 User, 5 Ticket, 5 DataScraper, 3
```

```
DataPipeline, 5 RequestPipeline

check NoNotificationHighCriteria

check NotificationLowCriteria
```