Goal: designing an app like Uber.

Scenarios of the system (use-cases, functional requirements, ...):

- The passenger selects a destination. The server computes the path cost according to the passenger location (passenger source), the traffic between source and the destination, weather, current time of the day, number of destinations (common: 1, 2), halt time in secondary destinations, and some other criteria. After server determines the trip cost, it sends the passenger's request to drivers nearby close to passenger. The trip begins when a driver accepts the passenger's request,. If no driver do so within 5 minutes, then the request is canceled and the passenger should try again.

- How to define the nearby? For simplicity, in initial version, assume that passengers within 2km distance are shown to the driver.

- The passenger must be able to cancel the request if no drive has already accepted the request.

- The trip has two phases. The first phase begins with the driver approval and ends when the passenger gets in the car. The driver must hit a button (trip start button) to inform the server and the passenger that the trip is actually started, i.e, phase one has been finished and phase two is started. Then the second phase begins. The second phase ends when the passenger reaches the destination.

- The passenger must be able to cancel the trip in the first phase.
- The driver must be able to cancel the trip in the first phase.
- The passenger cannot cancel the trip in the second phase. (?)
- The driver cannot cancel the trip in the second phase. (?)

- What happens if the passenger doesn't enter the car while the driver has already arrived at the passenger location? In this case, the driver can cancel the request. What about banning the passenger after many cases of such an event?

- What happens if the driver doesn't come to the passenger location? In this case, the passenger can cancel the request. What about banning the driver after many cases of such an event?

- The driver must inform the server that the trip is ended at the end of the trip by clicking a button (trip finished button).

- What happens if the driver announce that the trip is started while the passenger is not in the car? What if the driver then says that the trip is ended then receive the money without making the trip? In this case, the operator must solve the issue by assessing the passenger location and the driver location. (class diagram?)

- What if the driver doesn't hit the button between phase one and phase two while the passenger is in the car? Answer: The driver looses money if doesn't do so. No action is required.

- What happens if the driver doesn't hit the button (tell the server that the trip is over) at the end of the trip? Answer: The driver looses money if doesn't do so. No action is required.

- What if the driver avoid to drive to the passenger destination and then hit the button that says the trip is over while has not reached the destination? Answer: The passenger calls the operator and the operator either cancels the trip or the passenger is lying and nothing happens. This can be done by comparing the location of driver with destination. (class diagram?)

- After the end of the trip, price cost is debited from the passenger. The cost mines the fee is credited to the driver.
    - If the passenger doesn't have enough credit to pay for trip, the driver must either receive cash from the passenger at the end of the trip (in this case, the fee is reduced from driver credit. The driver credit may become negative) or the passenger credit will become negative. This choice is up to the passenger.
    - The passenger credit cannot reduce below a specified threshold set by server.
    - The fee can be zero.
    - The passenger can use discount code if has any. Discount code is valid if not expired, related to the passenger, not used yet, and some other criteria holds.
    - Operator must be able to generate discount codes.
    - Operator can customize fee per driver.

- After the end of the trip, the passenger can rate the driver. Also, the driver can rate the passenger.

- The passenger can reserve a trip for future. (class diagram?)

- The passenger can credit the wallet.

- The driver can withdraw wallet within 24 hours delay.
    - cannot withdraw negative wallet.

- A recommender system is needed to recommend a new trip based on the passenger's previous trips. (client side class diagram?)

- The passenger must be able to assign two consecutive destinations. The driver must visit the first destination first, then the second (last) one.

- The passenger must be able to assign halt time in the first destination in the case of two destination kind of trip.
- What if the driver prevents to carry the passenger to the first destination and just carry to the second destination?
- What if the driver don't wait for the passenger in the first destination, i.e, bypass halt time?
- What if the passenger spend more time than what specified by the halt time?

- Passenger and Driver must be able to see the trips history.

- Passenger and driver must be able to export trip information into pdf.

- Operator must have a panel to monitor current state of the system including but not limited to current trips. (class diagram?)
- Other features of panel

- Driver constantly receives new requests from passengers nearby unless accepts one. Then, the driver doesn't receive new requests unless cancels the current accepted request or finishes the trip. (client side class diagram?)

- After a driver registers into the system, his/her information is first gets verifid by the system. If the police API approved the driver, the driver can receive passenger requests. Otherwise, no request is sent to the driver.

- Passenger must be able to register in the system by providing username, phone number, name (optional), password (optional) (If no password is set, system generates one time password(OTP)).

- The path the driver rides must be recorded in the server so that the operator can later use the path. This holds for passengers too.

- The operator has the right to cancel a trip even it's already started.

- Admin Panel: Develop an admin panel to monitor the system, manage user accounts, handle support requests, and generate discount codes. (class diagram?)

- Passenger and driver can login.

- Passenger and driver can fill profile.

- Passenger must be able to select source and destination location by searching places. (client side class diagram?)

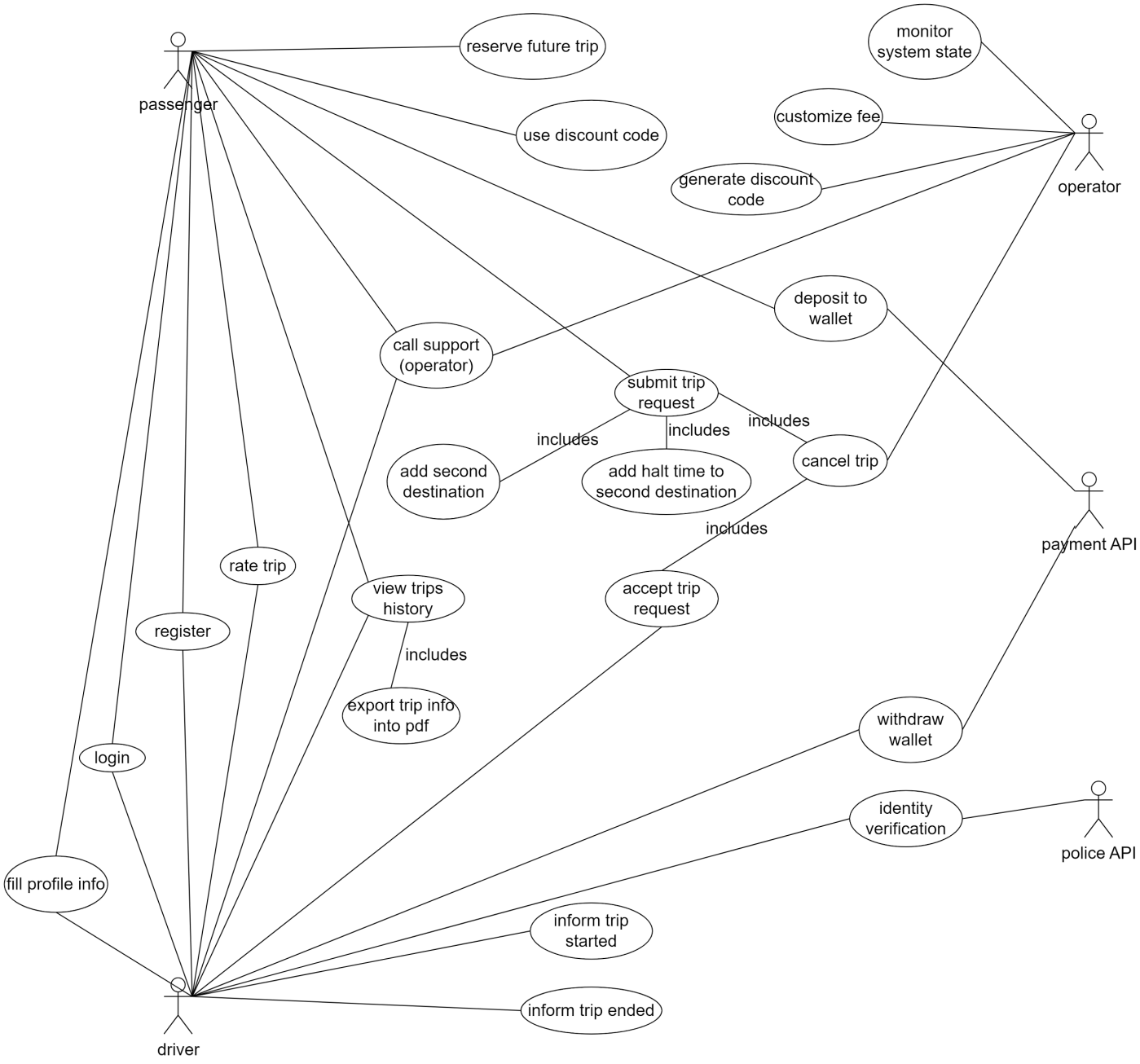
- chat: between passenger and driver (client side class diagram?)

System reqs:

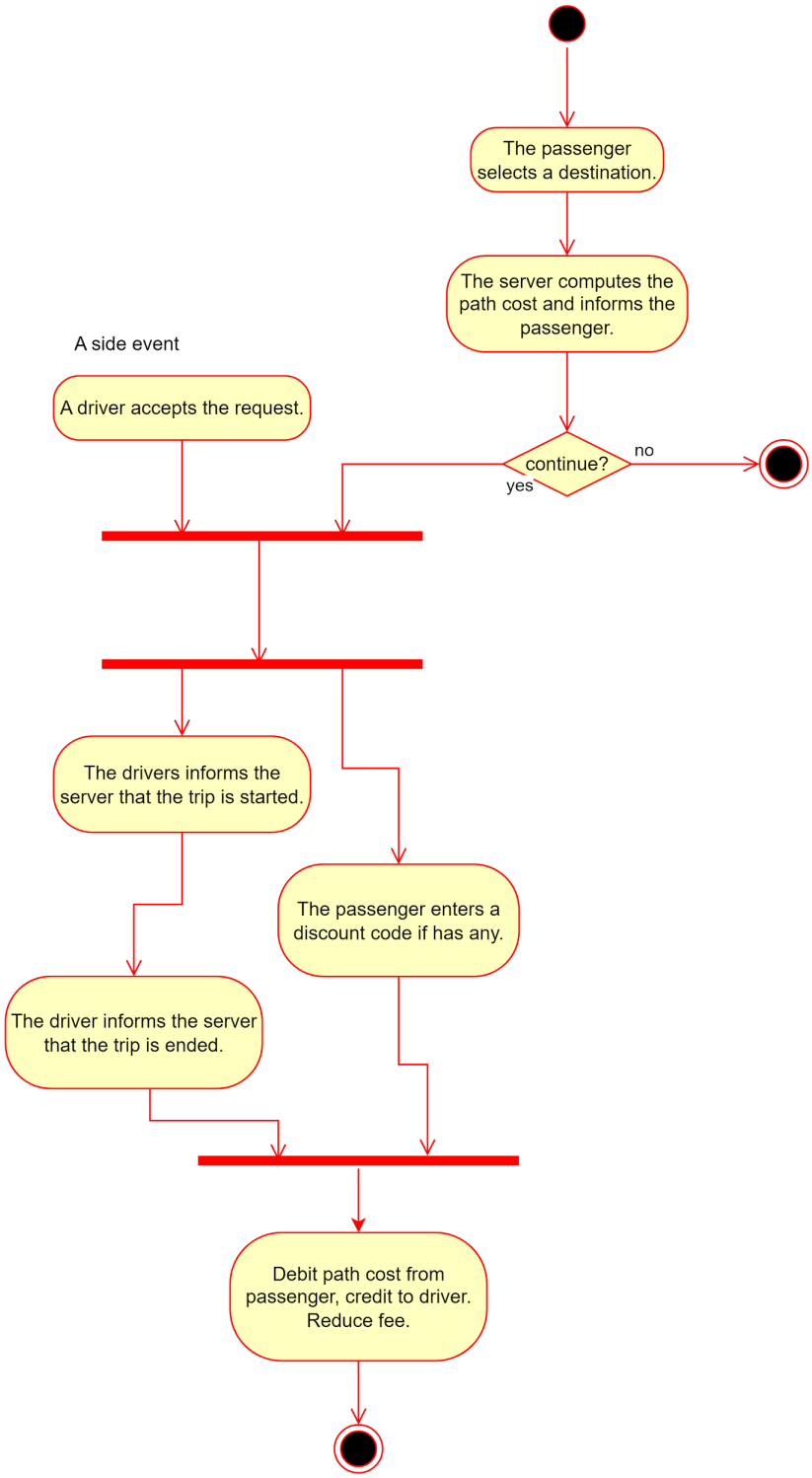
security, scalability, reliability, performance, fault tolerant, ...

---

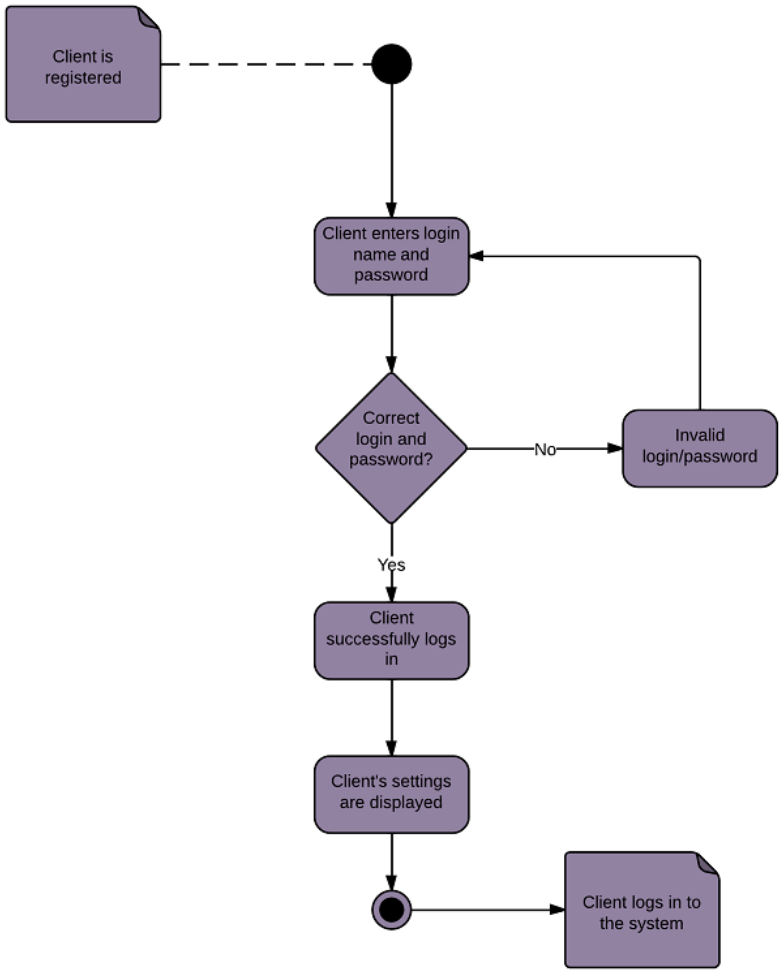
use case diagram:



Activity diagram:



It is also possible to include other activity diagrams too. For example, a login activity diagram looks like below:

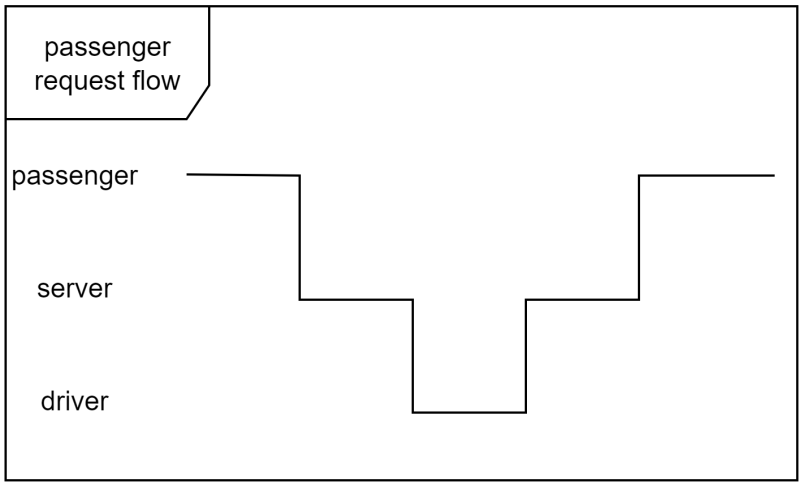


ref: [1]

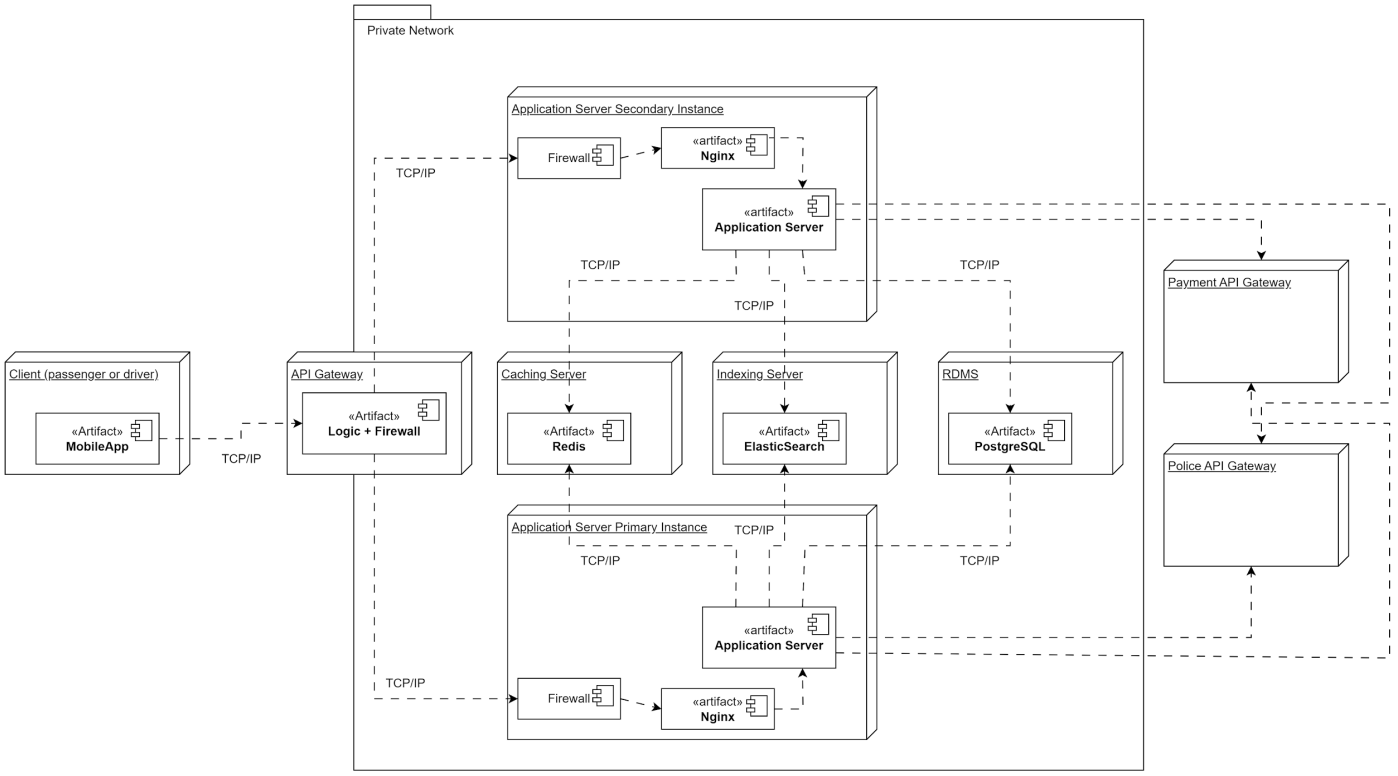
I decided not to include more activity diagrams because those are just too much simple flows.

Timing diagram:

[1] https://www.lucidchart.com/pages/uml-activity-diagram



Deployment Diagram:

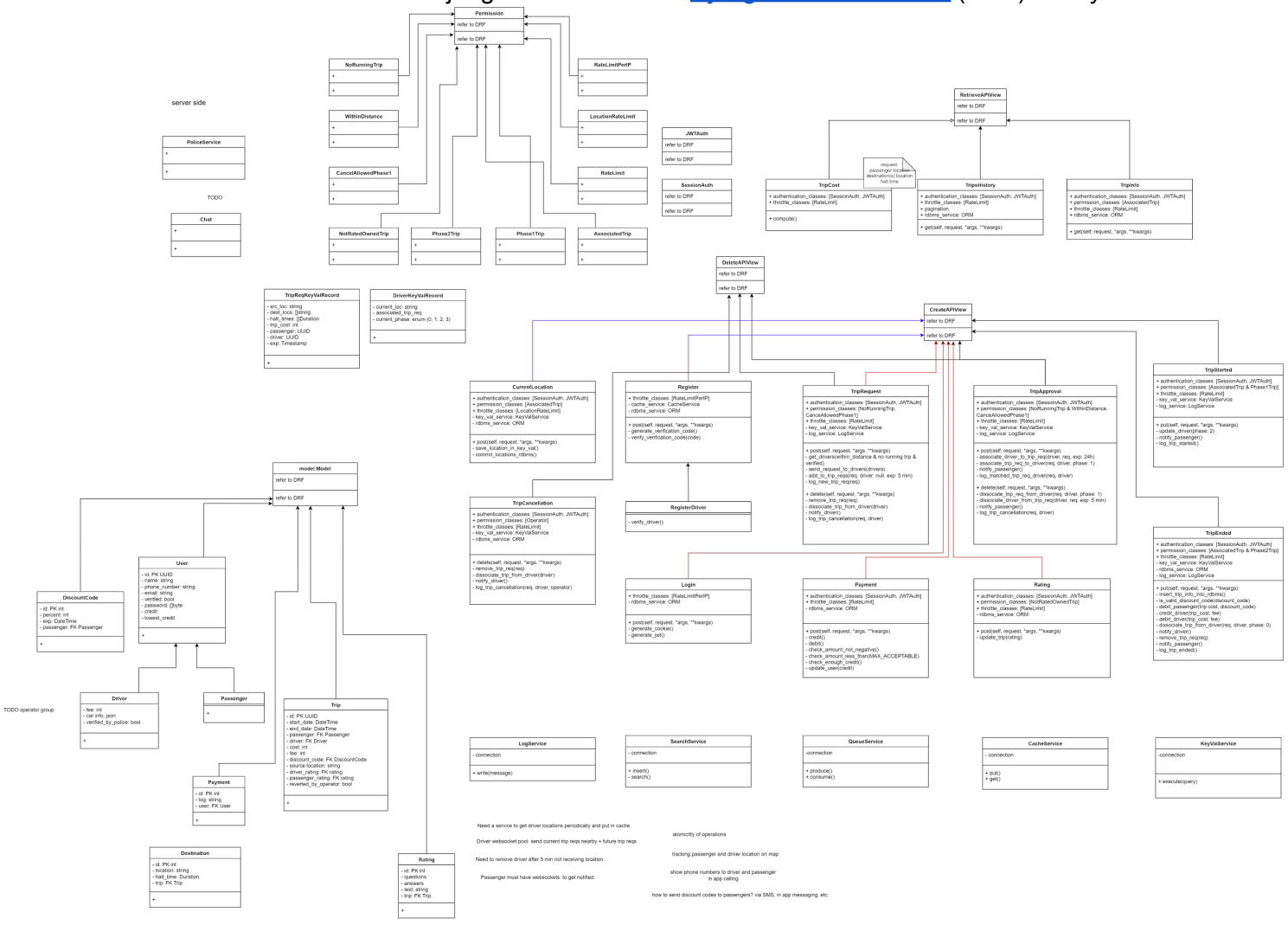


Artifact symbol should have changed to something like this one:

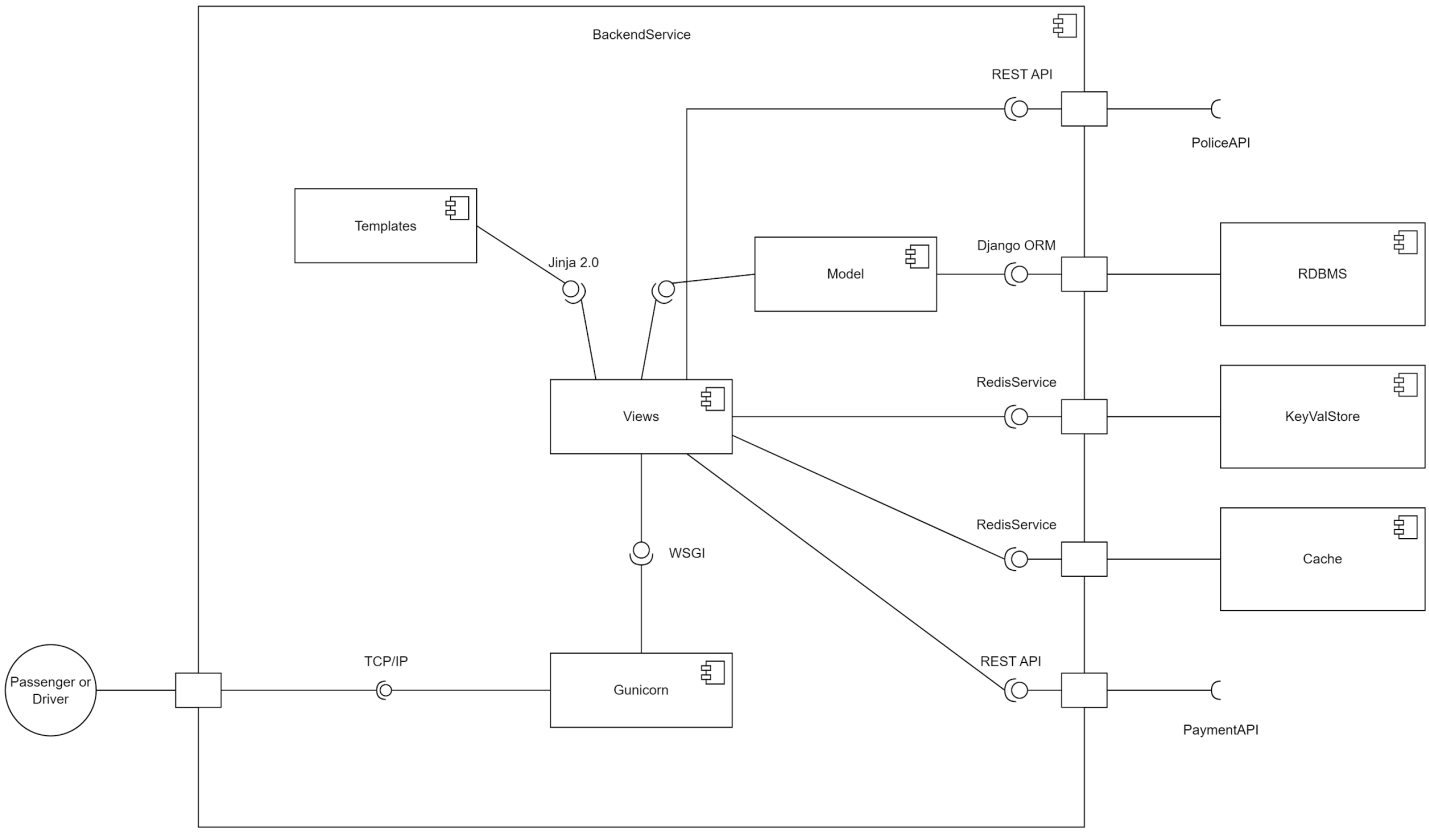
We can also easily include message queues to the deployment diagram like Indexing Server.
Note: Indexing, caching, relational database, and message queues are replicated in their owns. This is not shown in the deployment diagram.

Class Diagram:

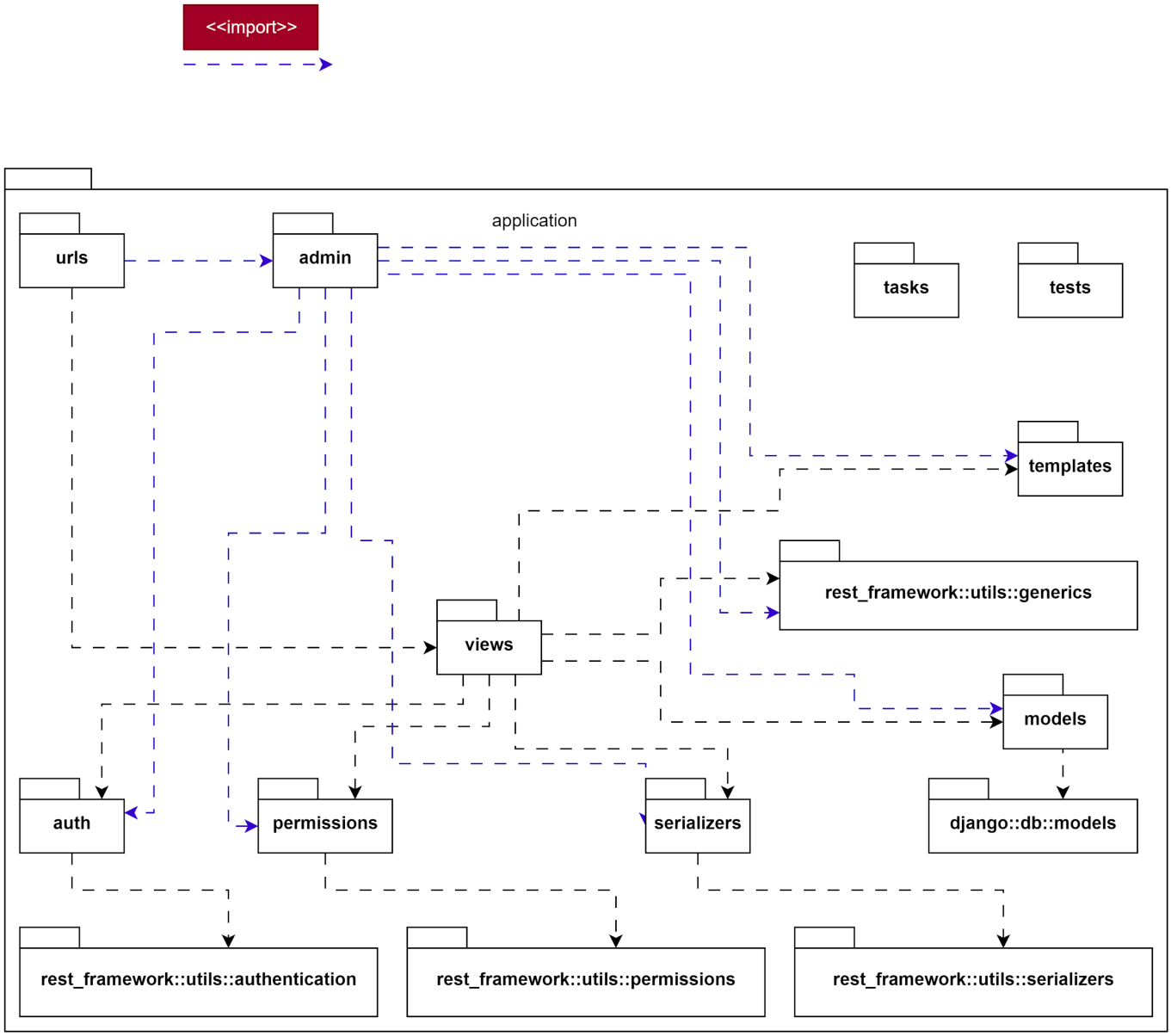
For simplicity, associations (aggregation, composition, and dependency arrows) are ignored. Inheritance arrow heads must be displayed as empty but here in the diagram they're shown as full heads.
The UML is created based on the Django framework and DjangoRestFramework (DRF) library.



Component Diagram:



Profile Diagram:


<<import>>
application
urls
admin
tasks
tests
templates
rest_framework::utils::generics
views
models
auth
permissions
serializers
django::db::models
rest_framework::utils::authentication
rest_framework::utils::permissions
rest_framework::utils::serializers

سند ریسک:

1. **ریسک: عدم پذیرش مشتریان توضیح: وجود رقبا و عدم تضمین دریافت موافقت مشتریان جدید ممکن است باعث کاهش تقاضا و کمبود مشتریان برای سرویس تاکسی اینترنتی شود.**
2. **ریسک: نقص فنی در نرم‌افزار توضیح: وجود باگ‌ها، عملکرد نامناسب سامانه و نقص‌های فنی می‌تواند باعث بروز مشکلات فنی، نارضایتی کاربران و از دست دادن اعتماد آن‌ها شود.**
3. **ریسک: مشکلات حفظ امنیت و حریم خصوصی توضیح: حملات سایبری، نفوذ به سیستم، نقض حریم خصوصی کاربران و رسیدن به اطلاعات حساس مشتریان می‌تواند عواقب جدی امنیتی و قانونی را به همراه داشته باشد.**
4. **ریسک: مشکلات مرتبط با رانندگان توضیح: رانندگان ناشایسته، نارضایتی رانندگان از سیستم، تأخیر در تحویل سفر و نارسایی تعداد رانندگان ممکن است به مشکلاتی در ارائه سرویس منجر شود.**
5. **ریسک: تغییرات قوانین و مقررات توضیح: تغییر در قوانین حمل و نقل و مقررات مربوط به تاکسی اینترنتی، الزامات جدید امنیتی و مالی و تأثیرات آن بر سیستم و کسب و کار ممکن است نیاز به تغییرات و سربارهای اضافی داشته باشد.**
6. **ریسک: رقابت با سایر نمونه‌های مشابه توضیح: وجود رقبا و سرویس‌های مشابه تاکسی اینترنتی که رقابت شدیدی را در بازار ایجاد می‌کنند، ممکن است به کاهش تقاضا، تنگنا در بازار و کاهش سهم بازار منجر شود.**
7. **ریسک: مشکلات مرتبط با پرداخت و تسویه حساب توضیح: مشکلات در پرداخت و تسویه حساب با رانندگان و مشتریان، ممکن است به تأخیر در تسویه و پرداخت، نارضایتی و از دست دادن اعتماد مشتریان و رانندگان منجر شود**
8. **ریسک: تغییرات فناوری توضیح: پیشرفت‌های فناوری، تغییرات در سیستم‌ها و زیرساخت‌ها ممکن است نیازمند تغییرات و به‌روزرسانی‌های مداوم در پروژه تاکسی اینترنتی باشد**
9. **ریسک: عدم قابلیت اطمینان و عملکرد نامناسب سرویس توضیح: از دست دادن ارتباط شبکه، خرابی سیستم‌ها، کاهش عملکرد سرویس و عدم قابلیت اطمینان ممکن است به نارضایتی کاربران، از دست دادن مشتریان و خسارات مالی منجر شود**
10. **ریسک: مشکلات مرتبط با تقسیم‌بندی مناطق توضیح: مشکلات مرتبط با تقسیم‌بندی و مدیریت مناطق، اختلافات قوانین و مقررات محلی و مشکلات جغرافیایی ممکن است به مشکلات در ارائه سرویس تاکسی اینترنتی در مناطق مختلف منجر شود**

11. ریسک: مشکلات مالی و مدیریت مالی توضیح: کاهش تقاضا، نارضایتی مشتریان، مشکلات مالی و عدم مدیریت مناسب منابع مالی ممکن است به مشکلات مالی و تأخیر در توسعه و پیشرفت پروژه منجر شود
12. ریسک: تغییر در نیازمندی‌ها و اولویت‌بندی‌ها توضیح: تغییرات در نیازمندی‌ها، اولویت‌بندی‌ها و تغییرات درخواست‌های مشتریان ممکن است نیازمند تغییرات در پروژه و تأخیر در ارائه خدمات باشد

سند ویژن:

ویژگی‌های اصلی:

1. رزرو و سفارش آسان تاکسی از طریق اپلیکیشن موبایل: کاربران قادرند به راحتی و با چند کلیک سفر خود را رزرو و تاکسی مورد نظر خود را سفارش دهند.
2. امکان پیگیری و مشاهده موقعیت تاکسی در زمان واقعی: کاربران می‌توانند در هنگام انتظار برای تاکسی، موقعیت دقیق تاکسی را در نقشه مشاهده کنند و زمان دقیقی که تاکسی به آن‌ها می‌رسد را ببینند.
3. ارائه سرویس تاکسی با رانندگان حرفه‌ای و مجرب: تمامی رانندگان متعهد و حرفه‌ای هستند و توانایی ارائه سرویسی امن و دوستانه را دارند.
4. سیستم امنیتی برای حمایت از مشتریان و رانندگان: سیستم امنیتی پیشرفته‌ای برای حمایت از اطلاعات کاربران و رانندگان، و جلوگیری از هرگونه سوءاستفاده و نقض حریم خصوصی فراهم است.
5. پرداخت آسان و امن از طریق برنامه: کاربران می‌توانند با استفاده از روش‌های پرداخت مختلف، به راحتی و امنیت کامل هزینه سفر را پرداخت کنند.
6. امکان ارائه نظرات و امتیازدهی به رانندگان و سرویس: کاربران می‌توانند پس از پایان سفر، نظرات و امتیاز خود را درباره راننده و سرویس ارائه شده ثبت کنند تا بهبود و بهتر شدن کیفیت سرویس را تشویق کنند.
7. پشتیبانی مشتریان 24/7: تیم پشتیبانی پروژه در دسترس 24 ساعته است و کاربران هر زمان که نیاز به راهنمایی یا حل مشکل دارند، می‌توانند با آنها تماس بگیرند و دریافت کمک کنند.

مشتریان هدف:

1. **افرادی که به دنبال راحتی و سرعت در سفرهایشان هستند: افرادی که نیاز دارند به سرعت و راحتی به مقصد خود برسند و از مزایای استفاده از تاکسی اینترنتی برخوردار شوند.**
2. **افرادی که نیاز به رزرو آسان تاکسی دارند: افرادی که می‌خواهند قبل از سفر خود را رزرو کنند و از نگرانی درباره یافتن تاکسی در زمان مورد نیاز خود مصون باشند.**
3. **رانندگان تاکسی که به دنبال فرصت شغلی بهتر هستند: رانندگان تاکسیی که علاقه‌مند به فرصت شغلی بهتر هستند و می‌خواهند با ارائه سرویس در پلتفرم تاکسی اینترنتی، درآمد و موقعیت شغلی بهتری داشته باشند.**

چشم‌انداز کلی: تاکسی اینترنتی آنلاین با ارائه یک پلتفرم معتبر و قابل اعتماد در حوزه حمل و نقل شهری، به مشتریان امکان می‌دهد تا با راحتی و اطمینان، به مقصد خود برسند. هدف ما ارائه تجربه‌ای خوشایند، سریع و امن برای مشتریان و رانندگان است. ما بر روی کیفیت سرویس، رضایت مشتریان و حفظ اعتماد آنها تمرکز داریم و به عنوان یک پلتفرم مورد اعتماد در بازار تاکسی اینترنتی شناخته می‌شویم

To scale the system:

horizontal scaling:

1. replacate web server instances and code in an stateless style in each web server. share state through databases specially Redis.
2. scale databases, replication and sharding
3. break monolith to microservices:
    a. seach service backed by ElasticSearch
    b. payment service
    c. API gateway: firewall, routing, load balancing, service discovery
    d. static content servers
    e. remained business logic in a service
4. replicate each microservice
5. dedicated team to manage Kubernetes infrastructure.
6. dedicated team for data engineering including setting up Hadoop ecosystem for storage and processing (streamming, batch) and Spark.
7. edge load balancers
8. setting up in two different datacenters: primary and secondary
9. use Kafka as message queue