Mobile Phone App Specification

Brief Introduction:

The app on the mobile phone is used to receive commands from the server. It also sends the GPS location to to server.

When the driver gets the phone, he will need to log into the system. After logging in, the server will know that one vehicle is available. The vehicle manager in Server application will assign a route to this vehicle. The driver will get the task set for current shift. If the plan changes, he will be informed immediately. The planned route is 12 hours long (or ends at 8 o'clock AM/PM). After all tasks finished. The driver can either log off *or choose to continue work*. (This is for drivers who are willing to work for more than 12 hours.)

Other Requirements:

When offline, the driver should still be able to set the status of a task. The status of tasks will be uploaded to server immediately when connection is restored.

The drivers are asked to manually set the status of a task. But when GPS location shows that he is within the range of a port for a certain amount of time, the server should inform him to set the status of the task he is currently doing.

UI Design:

Function calls:

login(String id, String VehicleID) returns *Session*

This function will send a log in request to server. The server will then return a *Session* to the mobile app, so that the log in state will be kept even the phone is offline.

If a user tries to log in again without logging off (this may happen if the phone was turned off in the middle of action), this function will match the token with the one on the server. If the session IDs are the same, the driver will continue using the *Session* he got before. The driver will be informed that he is continuing from previous work. If the session IDs are not the same, driver will get a new Session.

If a session is not found on mobile phone but available on server, then the mobile phone will get a session copy from the server.

The reason of using *Session* is that the driver may extend their work to one day, and they may forget to logoff. The *Session* will expire if certain criteria meet.

logoff()

The log off request is sent to the server. When server replies, it will delete its current session. If there are tasks unfinished, prompt the user. The user can leave the tasks unfinished.

fetchTaskUpdate()

Task updates are fetched from server. When network is disconnected, the app should either show this status in the status bar or connect automatically. It will log in again with the same session if connect is made again. The task update is done using the “route” and “task” xml script. This method should be run periodically. For example, one time per minute.

changeTaskStatus(Task t, int status)

This function changes the status of a task. If the application is online, the status change will be submitted to server as well. If the mobile phone is offline and several tasks' status are changed, then this function should submit all changes after it is offline.

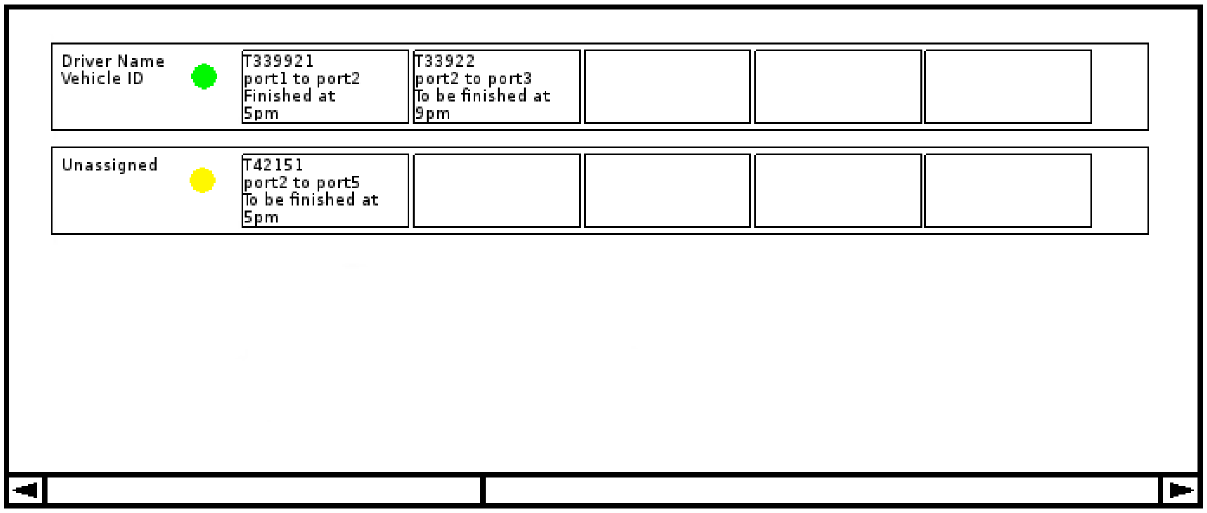
Server Specification

Brief introduction:

The server has following components: route operation UI, vehicle manager, GPS viewer, task viewer, task manager, communication manager and route planner.

Server Design:

1. Route Operation UI



It provides information for the whole route and every tasks. The *route frame* contains three parts.

The first part is the text label showing the current assignment of this route. In the example above, the driver name and vehicle ID is displayed.

The second part is the *route status indicator*. Green light means the routing plan is being carried out. After driver has logged in and got a session from server, the server automatically set this to green. Yellow light means the route is not assigned to any driver. Red light indicates that the session is expired, suspended or ended.

The third part is a container that contains multiple *task widgets*. For each task widget, the following information is displayed: Declaration ID, source port, destination port, estimated finish time or actual finish time.

Human Interaction

Menu items when route frame is right clicked:

1. GPS information.
2. Suspend session / Resume session

When “GPS information” is clicked, the **GPS viewer UI** pops up and the current vehicle is automatically selected.

When “Suspend Session” is clicked, a window should pop up to confirm this action. If the user selects “Yes”, the selected route's session will be suspended and the green indicator becomes red. The underlying function name is called suspend*Session()*.

“Really suspend session? You can enable it again before next session starts” “Yes”, “No”.

When the “Resume session” is clicked, the session will be resumed and the red indicator becomes green. The underlying function name is called *resumeSession()*.

Finally, if the same vehicle starts another session, the previous session will be ended. The underlying function name is called *endSession()*. The *route status indicator* will be turned off.

2. Vehicle Manager

The UI of vehicle manager is to be further discussed. It will be developed in next stage. The vehicle manager assigns vehicle in real world to a route in the algorithm. The drivers' information and vehicles' information are stored in database. The sessions are also recorded in database for future look ups.

Database for managing user accounts:

Table *Driver*

Driver Name (PK)
Password
Phone number

Table Vehicle

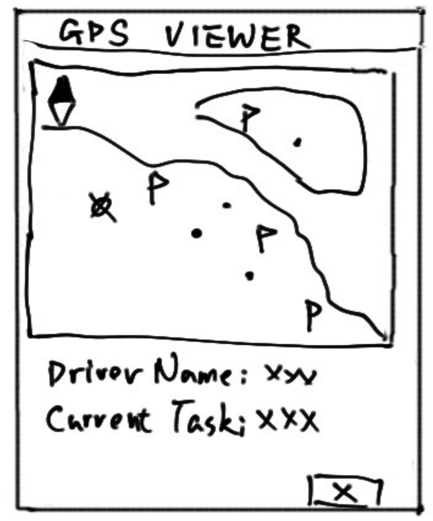
Vehicle ID

Table *Session* (makes a log for all sessions)

Driver Name (FK --- Driver[Driver Name])
Vehicle ID (FK --- Vehicle[Vehicle ID])
Start Time
Expire Time

Vehicle manager holds a copy of routes generated by the algorithm. Vehicle Manager can assign a route to a driver. Communication manager informs the vehicle manager about the updates of task. For example, task number 2 is completed with 2 small containers are transported. Then the corresponding task is modified the vehicle manager.

3. GPS Viewer (Need further confirmation from map provider.)



The **GPS Viewer UI** contains following elements: Map viewer and truck information viewer.

When the truck is clicked. Its information will be automatically displayed below the map viewer.

4. Task Viewer

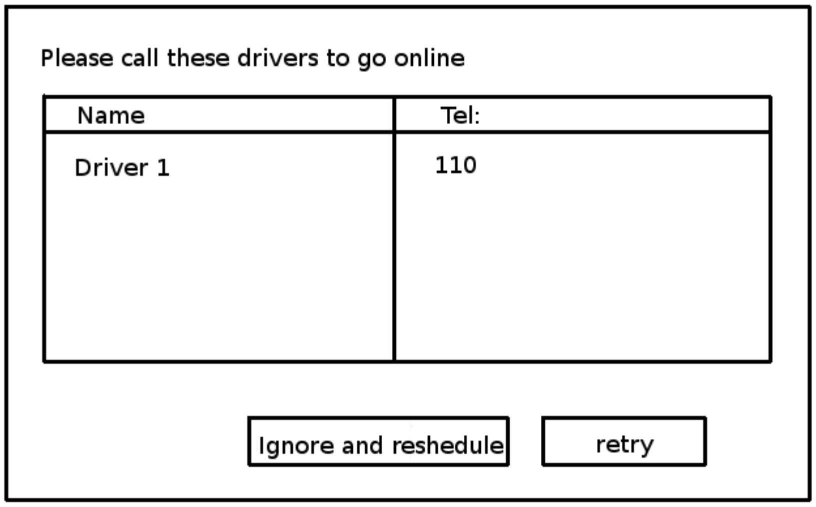
The **Task Viewer UI** shows all information for the selected task. Some fields' values can be changed (this is indicated by the third column of the table). The information comes from *task* (Defined in communication protocol section). After clicking “OK”, all things will be written to this task.

5. Task manager

Task manager maintains a pool of commodities. These commodities' information are stored in a database. Whenever new commodities are added or the pool is updated, the algorithm need to be restarted.

6. Communication Manager

How to update task plans?
Now we assume that the mobile phone already got tasks (Under this situation, the **vehicle manager** has assigned a route to this mobile phone). Whenever the algorithm finishes updating routes, it will try to submit routes to the communication manager. Then, the communication manager send updated routes to all drivers that have logged in and connected to the server. At this stage, the mobile phones will still use the old routing plans. After all mobile phones has confirmed that they have received the updated plans, the communication manager will then ask all mobile phones to use the new route plans. The whole process should use TCP to communicate.



If any one of these clients is failed to contact, the server should pop a window showing the list of these drivers with their telephone number. So that the operator can call them. He can also choose to ignore and reschedule. The communication manager should suspend these routes' session using suspendSession(Route r) and ask the algorithm to run again.

7. Other requirements (Please Ignore here now ^_^)

When the mobile app lost connection. The server will assume that the tasks will be carried out according to plan (or plus 1 hour late?).

receiveTaskStatus() has higher priority than *changeTaskStatus()* , Thus, *receiveTaskStatus()* will overwrite changes of *changeTaskStatus()* . If the operator want to change task status, he has to use the latest information provided by the driver.

8. Route Planner

when computer is free, run reschedule some time to improve solution.

TODO: thread implementation.

TODO: algorithm starter

TODO:

Class Definition:

1. Algorithm

Route

enum status = {UNASSIGNED, ASSIGNED , SESSION_SUSPENDED, SESSION_ENDED}	If a route is unassigned, algorithm is free to modify any part of it. If the route's session is suspended or assigned, all finished tasks will be frozen, the remaining tasks can still be modified by algorithm. If the route's session is ended, all unfinished tasks will be freed into task pool and removed from this route.
private Session s	The related session
public setSession(Session s)	Set the session for this route, also set r.status to “assigned”.
getTightness()	how much time the first task can be postponed without affecting the whole route plan. (inversed way of checking route).
Vector<Task> taskSet check()	Tasks in the route Checks itself for validity

Network

Vector<Vector<Route>> routesOfShifts	All routes, including future shifts.
public setVehicleQuantity(int n)	Set how many routes will be available today. It is NOT the number of drivers that logged in.
assignSession(Session)	Calls *r.setSession(Session s)*. The route will be chosen based on tightness of a route. That is, how much time the first task can be postponed without affecting the whole route plan.
startSession(Route r)	The *route status indicator* will be green.
suspendSession(Route r)	Set r.status to “session suspended”. The *route status indicator* will be red.
resumeSession(Route r)	Set r.status to “assigned”. The *route status indicator* will be green.
endSession(Route r)	Set r.status to “session ended”. The *route status indicator* will be turned off.
getCommodities()	Returns the commodity set.
addCommodity(c)	

RoutePlanner

run()	Run the algorithm from zero route plan.

reschedule()	Reschedule the current route
setCommodityPool(Vector<Commodity> commoditySet)	Sets the commodity pool for route planner. Must stop the route planner first.
setNetwork(Network nw)	Sets the network for the route planner.
setRoutes(Vector<Vector<Route>> routes)	In case this is the second run of routing planner. This can make the algorithm start with predefined routes
setBaseLineTime(Calendar c)	Sets the current time so all new tasks are considered to be started after this time

2. Vehicle Manager

Vector<Vector<Route>> routes	A copy of routes from the algorithm
Session validateLogin(DriverName name, Password p, VehicleID id)	This function will verify the name and password against the database. Once the driver is verified, it call calls network.*startSession(Session s)* and returns the session.
addTask(Task t, route r, int slot)	Adds a task from the task pool of Task Manager. It MUST inform the communication manager to send this update to driver. If the route check (route.check()) fails, you should give warning.
removeTask(Route r, int index)	Remove a task in a route. It MUST inform the communication manager to send this update to driver.
moveTask(Route r1, int index1, Route r2, int index2)	Move a task from one route to another. It MUST inform the communication manager to send this update to driver. If the route check fails (route.check()), you should give warning.

5. Communication Manager

ComManager

private sendTaskUpdateToClients()	Whenever the algorithm is run and the route is updated, all related drivers should be notified. This method does not notify the driver if the route plan is not changed for him.
public getTaskUpdateFromAlgorithm(Vector<Routes> r)	This method is to allow the algorithm to submit route update to communication manager. Only the changed route plans will be included.

private reportRouteUpdateCommunicationError()	When the communication manager fails to inform some of the clients, this method should pop up a window showing the list of these drivers with their telephone number. So that the operator can call them. If the operator choose to ignore, this method should set the related routes' session to “suspended” using suspendSession(Route r) and then reschedule the routes using reschedule() in RoutePlanner.
xmlParser()	Parse XML from clients.

6. Route Operation UI

What do you need from my algorithm?

7. Task Manager

addCommodity(Commodity c)	

Communication Protocol

The communication protocol is based on XML. The following tables are unified structure for both XML files and Java Classes.

Server to mobile

session

String driverName	Name of the driver
Calendar startDate	When this session is requested
Calendar expireDate	When the session should expire, if phone is offline, the app should inform the driver.
String vehicleID	License plate number

Task

String declar_ID	申报单号 declaration ID	Manual Modification	Mandatory in XML?
int sequenceNo	Sequence number n means it is the nth task in the route.	Yes	Yes
String src	Source port	No	yes
String dest	Destination port	No	yes
Calendar availableT	Available time of this task	No	yes
Calendar deadline	Deadline of this task	No	yes
int size	Size of the container	No	yes
int weight / String weight	Weight of the container (in tons) / or simply “heavy” ”light”	No	yes
int quantity	How many containers should be carried at once	Yes	yes
Calendar plannedStartTime		No	yes
Calendar plannedLoadTime		No	yes
Calendar plannedTravelTime		No	yes
Calendar plannedUnloadTime		No	yes
Calendar plannedFinishTime		No	yes
Calendar actualXXXTime	Actual values of planned time (So this item contains actually 5 items)	Yes	No
enum status = {planned, started, finished}		Yes	No

Route

Task tasks[]	Tasks for this route.

update

Task task	Which task is to be changed

Example of xml code

```
<route>
        <task>
                <declarID>T09090111</declarID>
                <sequenceNo>0</sequenceNo>
                ......
        </task>
        <task>
                <declarID>T09090112</declarID>
                <sequenceNo>0</sequenceNo>
                ......
        </task>
</route>
```

This xml is used when the server sends the new/updated route plan to the mobile phone.

```
<update>
        <task>
                <sequenceNo>0</sequenceNo>
                <declarID>T09090112</declarID>
                <status>finished</status>
        </task>
</update>
```

When task is completed, the mobile app will send this to server

```
<update>
        <task>
                <sequenceNo>0</sequenceNo>
                <declarID>T09090112</declarID>
                <weight>heavy</weight>
                <numberOfContainer>1</numberOfContainer>
        </task>
</update>
```

When the truck arrives at the port, the driver found that the small container is actually quite heavy. He will need to send this to the server.

It is very **important** that “update” should include tasks' declaration ID. The declaration ID is used as an verification method in case the server have different route plan. So this is for robustness purpose.

Task assignment for members

Li Jiaqi:

1. Mobile Interface
2. After logged in: just send a predefined route and session to mobile client now to check whether the XML parser on client and server is correct.
3. Communication manager.

Zhang Yao:

1. GUI of server side (route operation UI).
2. Define the interface that allows the server to interact with route operation UI. For example, which method will change the color of *route status indicator*. If the algorithm has finished generating routes, which method will be used to inform the UI that new routes are ready so that it can update the view.
3. GPS viewer 先不用急，等地图搞定了再说。

Zhu Haoyue: Database for vehicle manager and its class.

Chen Jianjun: DTD of XMLs. Modification of Algorithm data structure. Provide skeleton methods for server.

明天：大家演练一遍 svn 操作，设置好自己的 eclipse work space (其他也行)。切记切记。

然后我开始吧 skeleton method 都创建出来，其他成员思考 interface，下星期要弄出个 draft 来，和本文档类似。别人需要写什么 function 也写出来。比如张尧说 driver name 我不知道从那里弄来，建军要在 routes 里面提供这个接口。祝皓月的部分应该最早完成。现在密码之类的都明文传递，用来测试。所有的文本都用 UTF-8 !

算法设计，

shift starting time is variable and depends on the log in time of drivers. There is a system clock showing the current time.

What we do when 3 drivers are logged in and 5 more to be logged in? There should be another variable indicate the current number of available vehicles.

When the server assigns the route to a vehicle, it should give looser routes to vehicles to drivers who log in late. Loose route is a route that can be serviced later and still feasible.