پاسخ سوال6)

با توجه به اینکه، بررسی ما حول معماری سیستم نیازمند تحلیل نیازمندی‌های Nonfunctional است، در ابتدا به بررسی این همین نیازمندی‌ها و تعمیم دادن آن‌ها به ویژگی‌های سیستم می‌پردازیم.

نیازمندی‌های Nonfunctional متعددی قابل مطرح کردن هستند که در قالب یک دسته‌بندی کلی و مفهومی، به بررسی آن‌ها می‌پردازیم:

* Operational Requirements
* Performance Requirements
* Security Requirements
* Cultural and Political Requirements

Operational Requirement

این نیازمندی از تمام جنبه‌های تکنیکی، میزان integrity و ادغام‌پذیری و integrity سیستم و نهایتا maintainability قابل بررسی است. کاربرد برنامه‌ی هم ‌مسیر به‌گونه‌ای است که بر روی smartphone ها نیز جای‌گیرد و سخت افزار پیچیده ای نیاز ندارد. همچنین افزودن قابلیت نصب بر روی تمام سیستم عامل‌های معتبر نظیر Linux، Mac، Android مد نظر طراحان قرار دارد. همچنین برنامه‌ی همسفر نیاز به یک پشتیبانی مرکزی دارد.

Performance Requirement

از جنبه های Speed Requirement، Capacity Requirement و

Availability and Reliability Requirement قابل بررسی است. برنامه‌ی هم‌سفر تاکنون به گونه‌ای طراحی شده است که قابلیت Sync شدن با سرعت بالا و زمان Load پایین داشته باشد تا مورد استفاده‌ی کاربران قرار گیرد؛ همچنین پایگاه‌داده آن به صورت Real-time به روزرسانی می‌شود. پیشبینی تعداد کاربرانی که در هر لحظه از برنامه استفاده می‌کنند جز برنامه‌ی طراحان است، تا قابلیت سرویس‌دهی به کاربران به صورت گسترده فراهم شود.

Security Requirement

این نیازمندی از جنبه های System Value Estimate، Access Control Requirement، Encryption and Authentication و Virus Control قابل بررسی است. باید تخمینی از میزان درآمد هم مسیر به ازای واحدی از زمان محاسبه شود، هم چنین محدودیت‌ها و فعالیت‌های هر یک از کاربران در نقش‌های راننده یا مسافر یا ادمین، باید کاملا تمیز و مشخص شود.

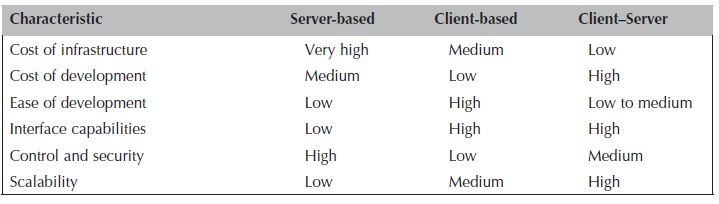
اطلاعات مربوط به هر کاربر نظیر شماره تلفن برنامه‌های شخصی و سفرهایش باید به درستی حفاظت شوند.

تمام فایل هایی که  برنامه اضافه می‌شوند باید بررسی شوند که حامل ویروس نباشند.

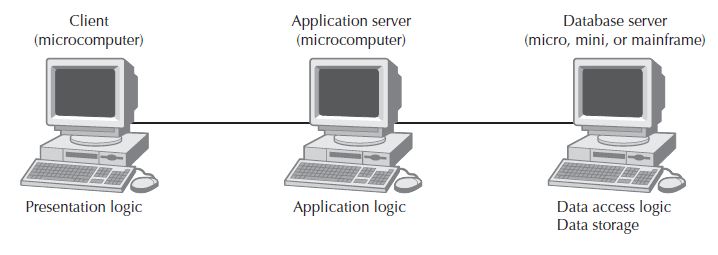
Cultural and Political Requirements

این نیازمندی از ابعاد Customization Requirement و Legal Requirement قابل بررسی است. برنامه هم مسیر باید نسبت به سیاست های کشور منعطف باشد و اگر برای مثال یکی از نهادهای دولتی لازم دانست که اطلاعاتی اضافه بر آنچه در نظرگرفته شده است درون پایگاه داده ذخیره شود، لازم به بررسی است.

حالا که با نیازمندی‌های مورد نظر سیستم آشنا شدیم، به مقایسه‌ی معماری‌های مختلف می‌پردازیم؛ جدول زیر به خوبی مقایسه ای را بین معماری ‌های مختلف طراحی نرم‌افزار انجام داده است:



با توجه به آنچه که در مورد نیازمندی‌ها ذکر شد، با توجه به اهمیت قابل توجه موارد Performance و Scalability و Security و همچنین گستردگی استفاده از سیستم در ابعاد بزرگ، بهترین انتخاب برای سیستم ما، یک معماری Client-Server است. گفتنی است که این معماری نیز امروزه به صورت سنتی و قدیمی خود مورد استفاده قرار نمی‌گیرند و پیشرفت چشم‌گیری داشته اند؛ امروزه معماری Multi-Tiered Client-Server دارای اقبال بیشتری است. در این دیدگاه، به نظر می‌رسد با توجه حجم درخواستی که برای سیستم در آینده متصور می‌شویم، یکی از بهترین انتخاب ها، معماری Three-Tiered Client-Server است. تصویری از این معماری در شکل زیر ارائه شده است:



همان طور که شکل فوق به خوبی نمایش می‌دهد، در این معماری، پایگاه داده در یک سرور جداگانه قرار داده می‌شود و کارهای مربوط به داده‌ها و ذخیره سازی داده در این سرور صورت می‌پذیرد؛ کارهای مربوط به عملیات اجرایی برنامه نیز در یک سرور جداگانه انجام می‌شود. این جداسازی سبب می‌شود که کارایی سیستم و سرعت آن تا حد قابل توجهی افزایش پیدا کند و سرویس‌دهی مجموعه در سطح بالاتری صورت گیرد. (گفتنی است که اگرچه این سیستم دارای کارایی بالایی است ولی این کارایی در حجم بالایی از تقاضا خود را نشان می‌دهد و در سیستم های کوچک که ارتباطات و کاربران محدودی دارد، استفاده از این سیستم به کارا نیست و سبب افزایش هزینه و دشواری پیاده‌سازی می‌شود.)