A. High-level block diagram design for the system:

```
[Database for tickets] <---> [System admin interface]
|
V
[User interface]
|
V
[On-line credit card validation system]
```

B. Function point analysis:

1. Count the number of function points:
   To count the number of function points, we need to determine the number of user inputs, user outputs, user inquiries, and user interfaces in the system.
   - User inputs:
     - The system admin interface allows the administrator to add and delete tickets from the database, so this counts as one user input. The user interface allows the user to select and purchase tickets, which counts as another user input. In total, there are two user inputs.
   - User outputs:
     - The user interface allows the user to view the available tickets and purchase confirmation, which counts as one user output. In total, there is one user output.
   - User inquiries:
     - The user interface allows the user to search for available tickets, which counts as one user inquiry. In total, there is one user inquiry.
   - User interfaces:
     - The system admin interface and the user interface both count as user interfaces. In total, there are two user interfaces.

   Using the IFPUG function point calculation method, the number of function points is:
   (2 * 3) + (1 * 4) + (1 * 3) + (2 * 4) = **14 function points**

2. Estimated effort based on function points:
   - Using the IFPUG function point estimation method, the estimated effort is:
     14 function points * 20 person-hours/function point = **280 person-hours**

3. Estimated LOC:
   - Assuming that the system is being developed in Java, we can use the average number of LOC per function point for Java to estimate the number of LOC for the system.
   - According to the IFPUG function point analysis manual, the average number of LOC per function point for Java is 30 LOC/function point.
   - Therefore, the estimated number of LOC for the system is:
     14 function points * 30 LOC/function point = **420 LOC**

4. Estimated effort using Jones' table:
   - Using Jones' effort estimation table, the estimated effort for a system with 420 LOC and a complexity factor of 2.5 (mid-range complexity) is: **18 person-months**

5. Reducing effort by 25%:
   - To reduce the estimated effort by 25%, we can consider the following options:

- Reduce the number of user inputs, user outputs, user inquiries, or user interfaces in the system by 25%. For example, we can remove one of the user inputs or user interfaces from the system.
- Use a higher level programming language or a more efficient development process, which can reduce the number of LOC required for the system.
- Split the system into smaller, independent modules and assign them to different development teams, which can increase the overall productivity and reduce the estimated effort.

6. Use more efficient algorithms or data structures in the system, which can reduce the number of LOC required to implement the desired functionality.