**Software Design Specification**

*Rahul Poddar (U101116FCS097)*

*Piyush Singhania (U101116FCS282)*

*Anubhav Paul (U101116FCS012)*

*Jatin Gupta (U101116FCS052)*

**The Software Design Specification Outline**

**1. Introduction**

The Software Design Specification Document is an enclosed documentation detailing the built structure of the software - present and future. In this document, the software built structure has been described using informative narratives and graphical representations, namely data flow diagram,use case diagrams, sequence diagrams and state diagrams and other pertaining models and diagrams.

**1.1 Purpose of this document**

The primary purpose of this document is to extensively describe the journey that the user undertakes to get the desired output from the input given. It also explains the processes undertaken by the admin to perform the actions required from the user, with all possible entries. The document details relationships between different classes and functions used in the application.

**1.2 Scope of the development project**

The primary motive of this application is to enable users to get access to the success stories of individuals and corporations - what success means to them and how they got there. This web app gives an opportunity for people to share their inspiring and informative journeys. Users are able to list their education, job, experiences, references, skills required and advice. The web-app also allows you to search for professionals by their industry and education. This app should be a ​one stop resources​​ for questions such as ​“What can I do with this major?”. “What skills will I need for this job?”. ”What training, certification, experience will I need to be successful?”

Our webapp is inspired from LinkedIn and other career development platforms.LinkedIn is one of the major social sites for career development, Our product does not just allow people to interact with each other but they can also share their success stories and get inspired through our platform. Safe to say, this has taken inspiration from LinkedIn but it’s a completely unique and stand-alone app and it cannot be seen as an extension of LinkedIn. It is completely different from LinkedIn as the other is a service primarily used by professionals and business persons to put up their own skills and achievements only for their self-promotion with little or no emphasis on helping fellow colleagues and budding entrepreneurs. However, Up the Ladder will serve as a third-party service wherein successful people’s stories and skill sets will be uploaded on the forum by us, with the sole purpose of connecting them with fellow students and professionals looking for guidance and inspiration.

**1.3 Definitions, acronyms, and abbreviations**

IEEE: Institute of Electrical and Electronics Engineers

SDS: Software Design Specification

**1.4 References**

IEEE SDS template

**1.5 Overview of document**

This SDS is divided into seven sections with various sub-sections. The sections of the Software Design Document are:

**1. Introduction:** describes about the document, purpose, scope of development project definitions and abbreviations used in the document.

**2. Conceptual Architecture/Architecture Diagram:** describes the overview of components, modules, structure and relationships and user interface issues.

**3. Logical Architecture:** describes Logical Architecture Description and Components.   
**4. Execution Architecture:** defines the runtime environment, processes, deployment view.   
**5. Design Decisions and Trade-offs:** describes the decisions taken along with the reason as to why they were chosen over other alternatives.

**6. Pseudocode for components:** describes pseudocode, as the name indicates.

**7. Appendices:** describes subsidiary matter if any.

**2. Conceptual Architecture/Architecture Diagram**

Conceptual Architecture is “Context” for the system’s use.

**2.1 Overview of modules / components**

This subsection will introduce the various components and subsystems.

**2.2 Structure and relationships**

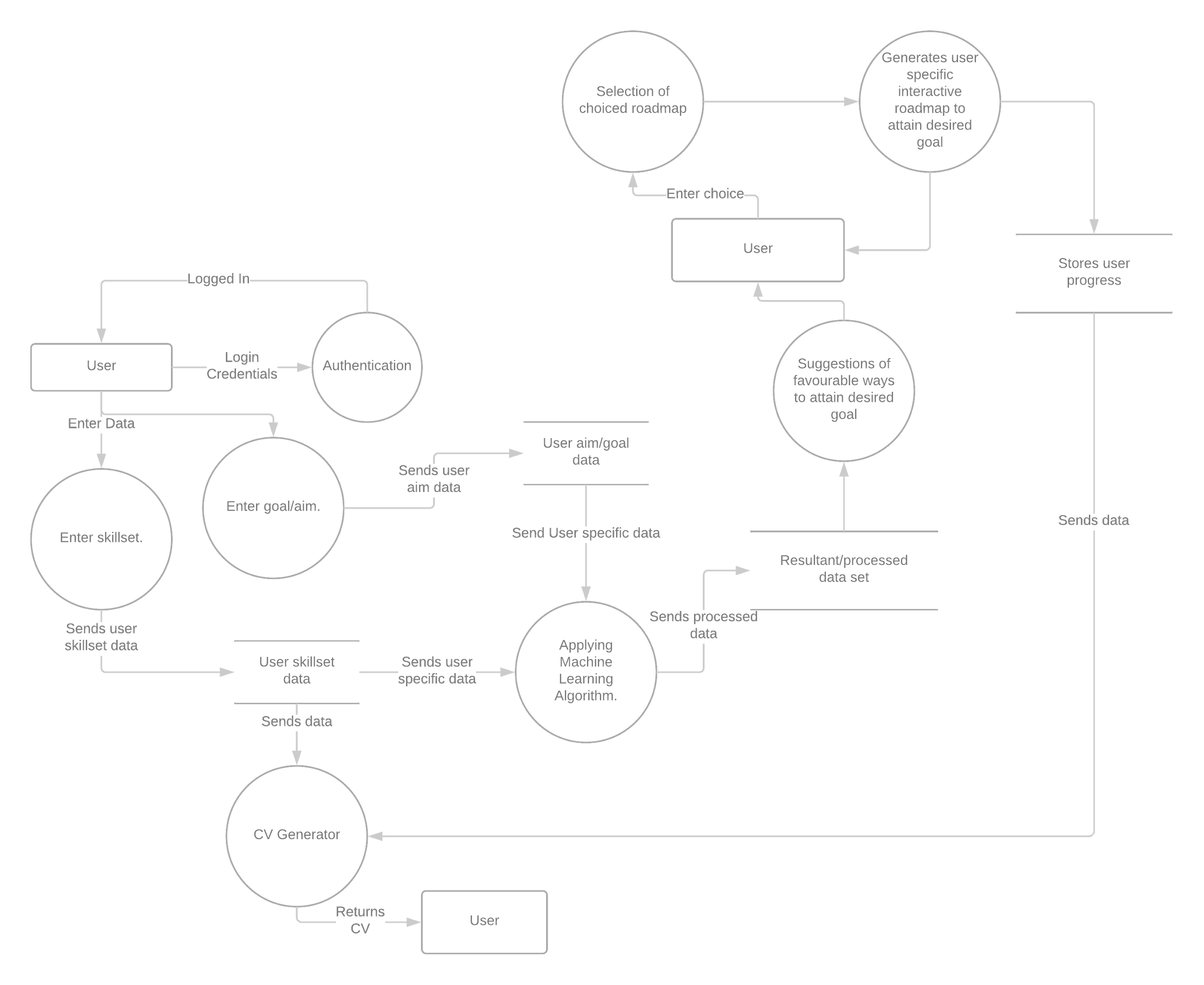
Make clear the interrelationships and dependencies among the various components. Structure charts can be useful here. A simple finite state machine can be useful in demonstrating the operation of the product. Include explanatory text to help the reader understand any charts.

**2.3 User interface issues**

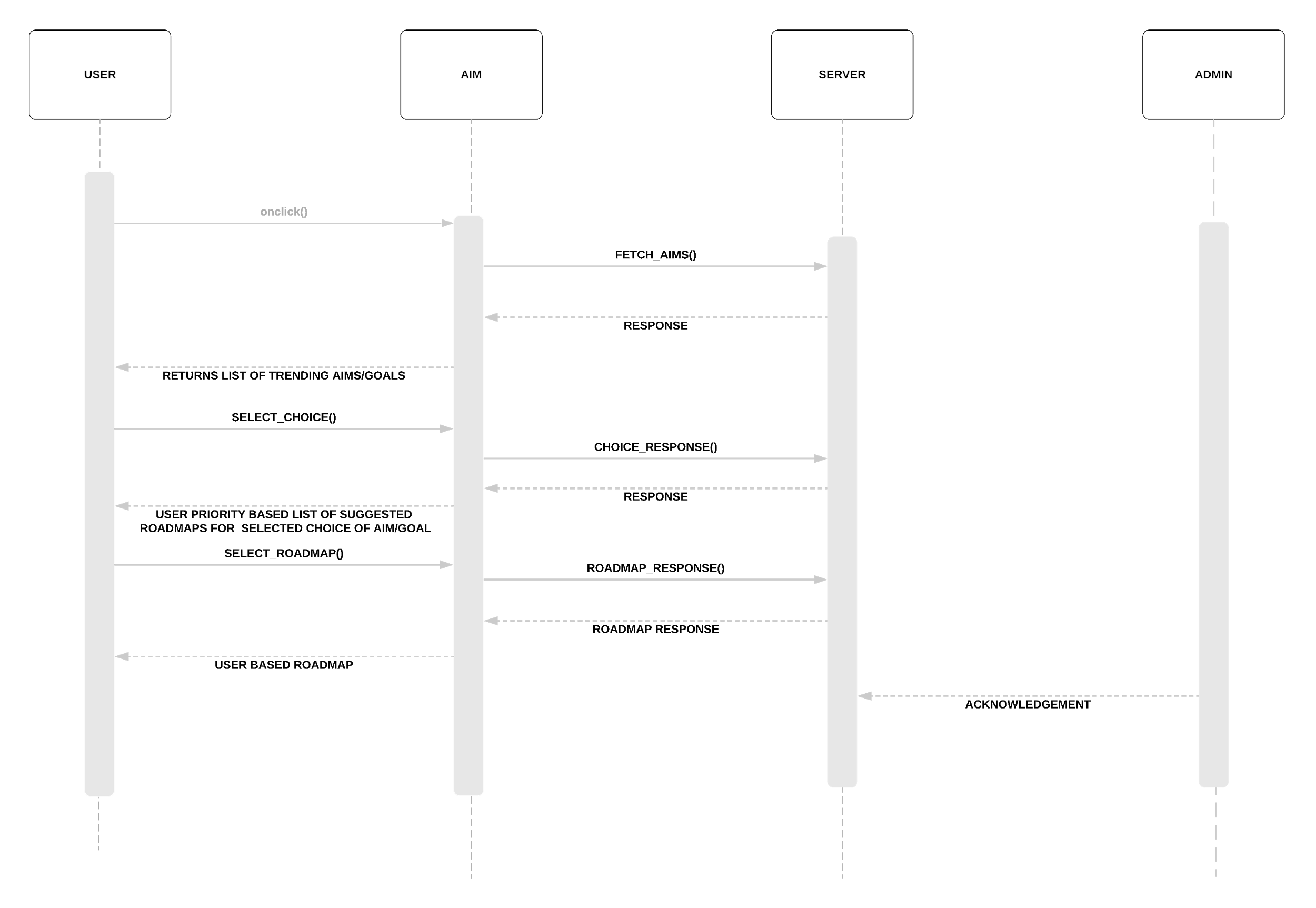
This section will present the main principles of the product's user interface. Use the personas defined in section 2.1 of your SRS to make specific examples. This section should not touch on technical details. You may want to include sketches and specific text messages.

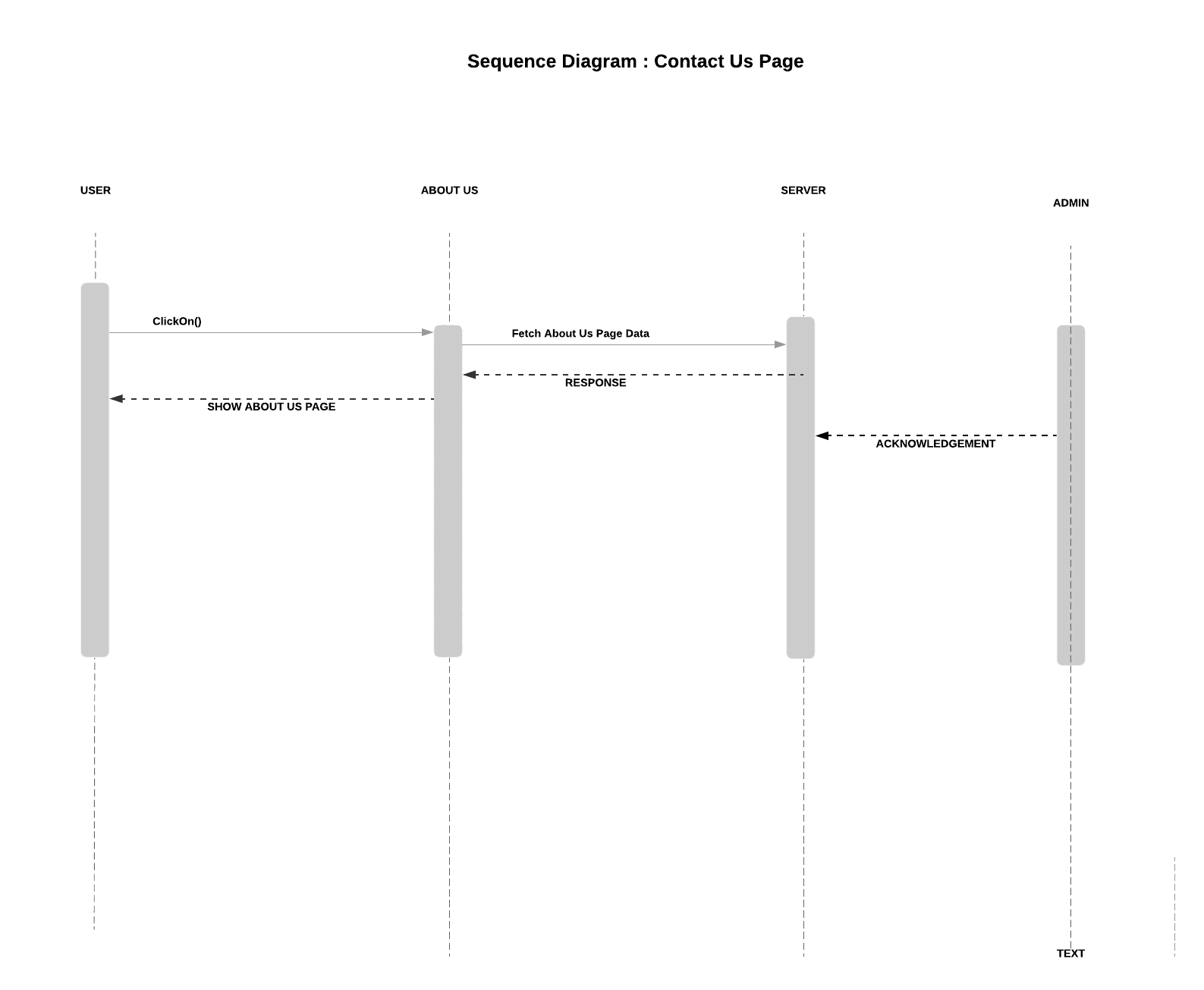
**3. Logical Architecture (Data Flow Diagram, Sequence Diagram, State Diagram)**

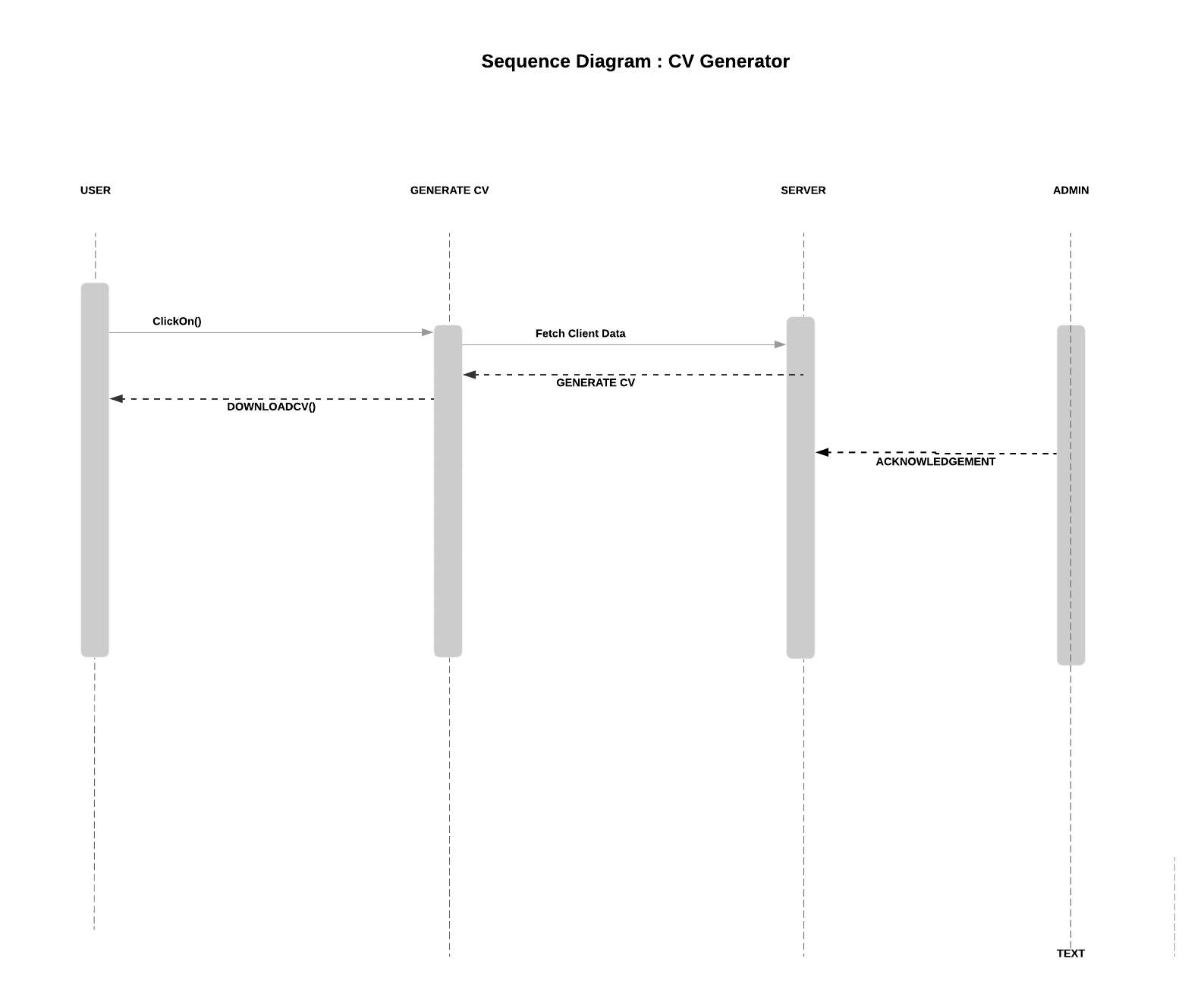
**Data Flow Diagrams**

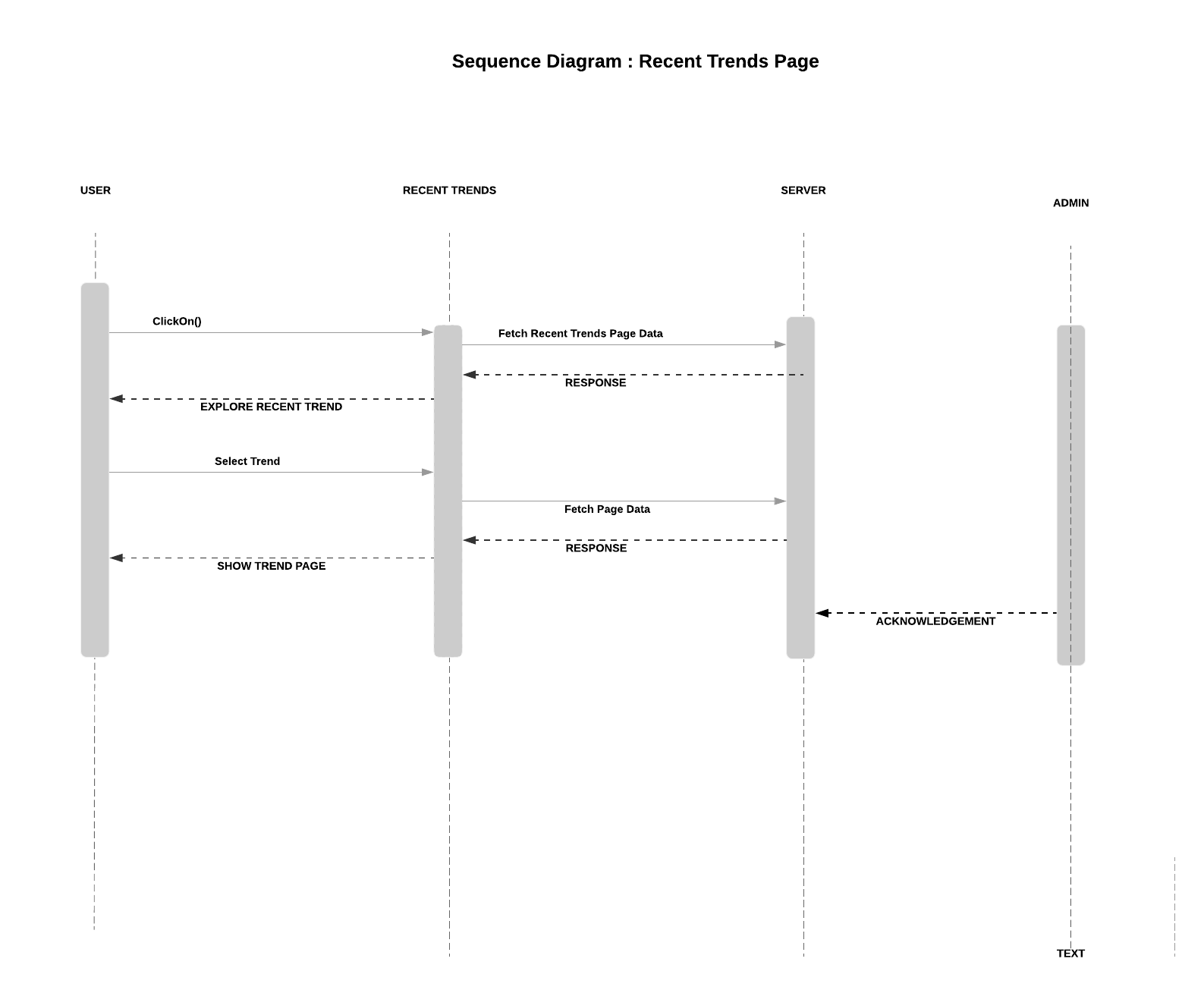


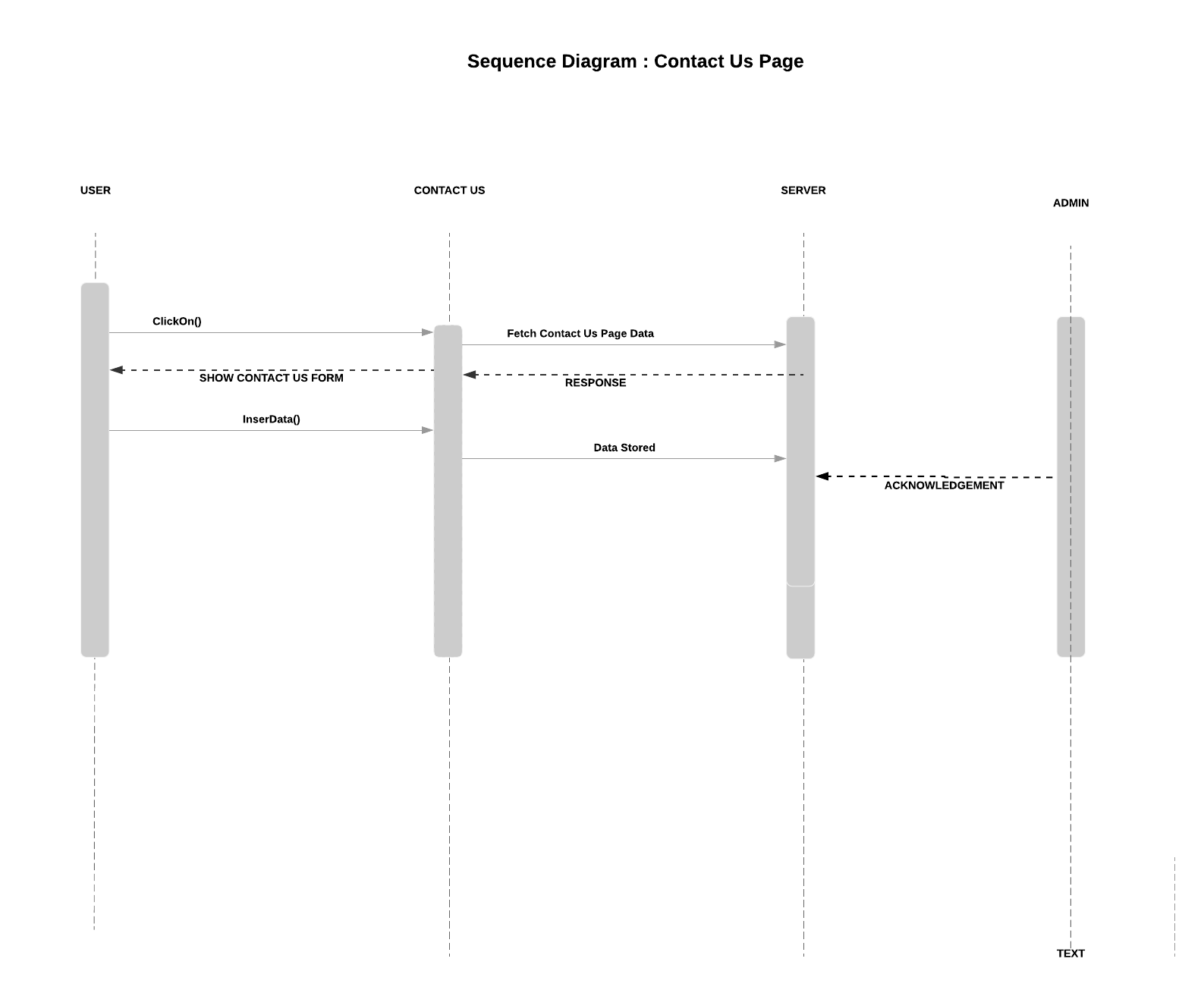
**Sequence Diagrams**

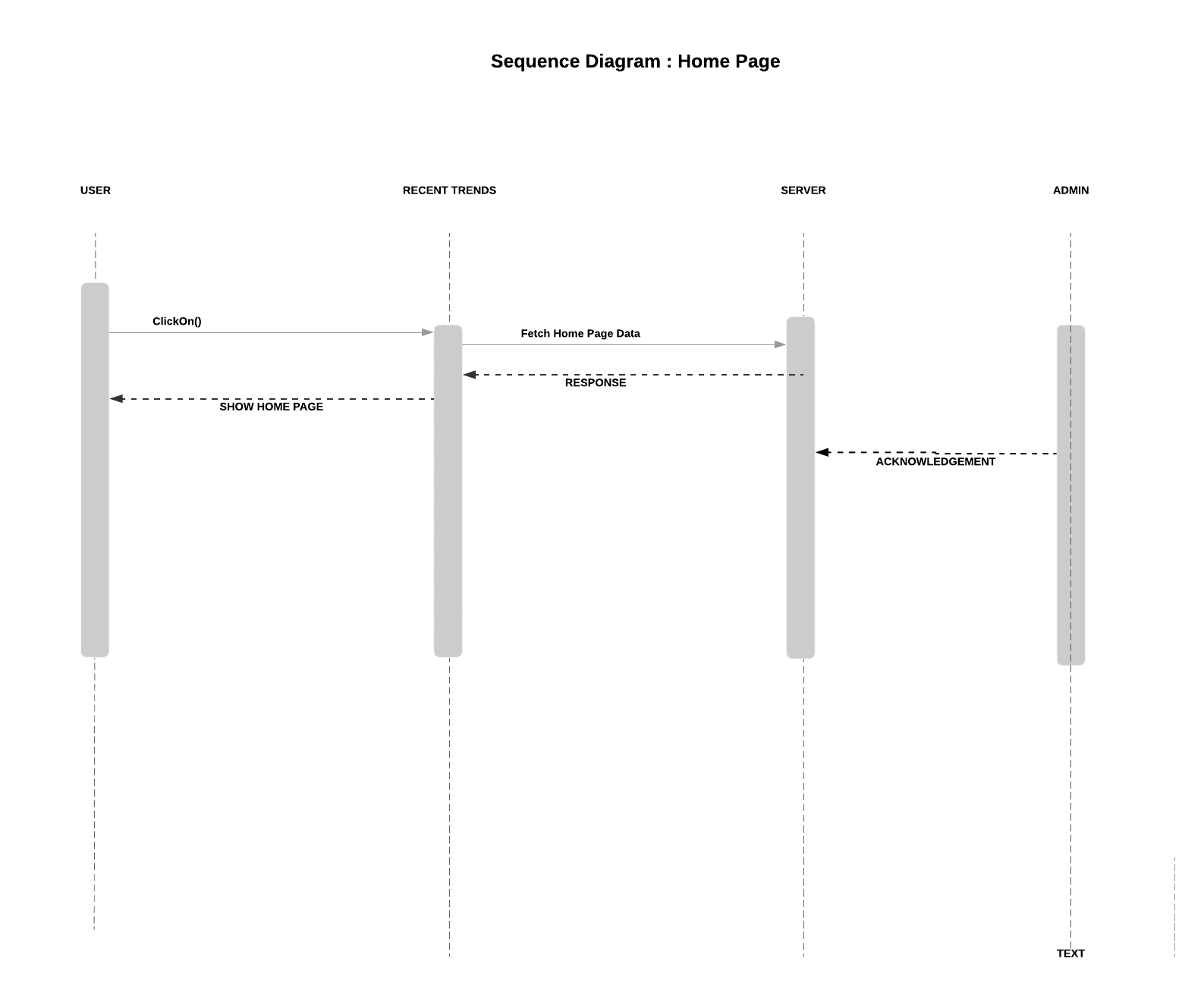
****

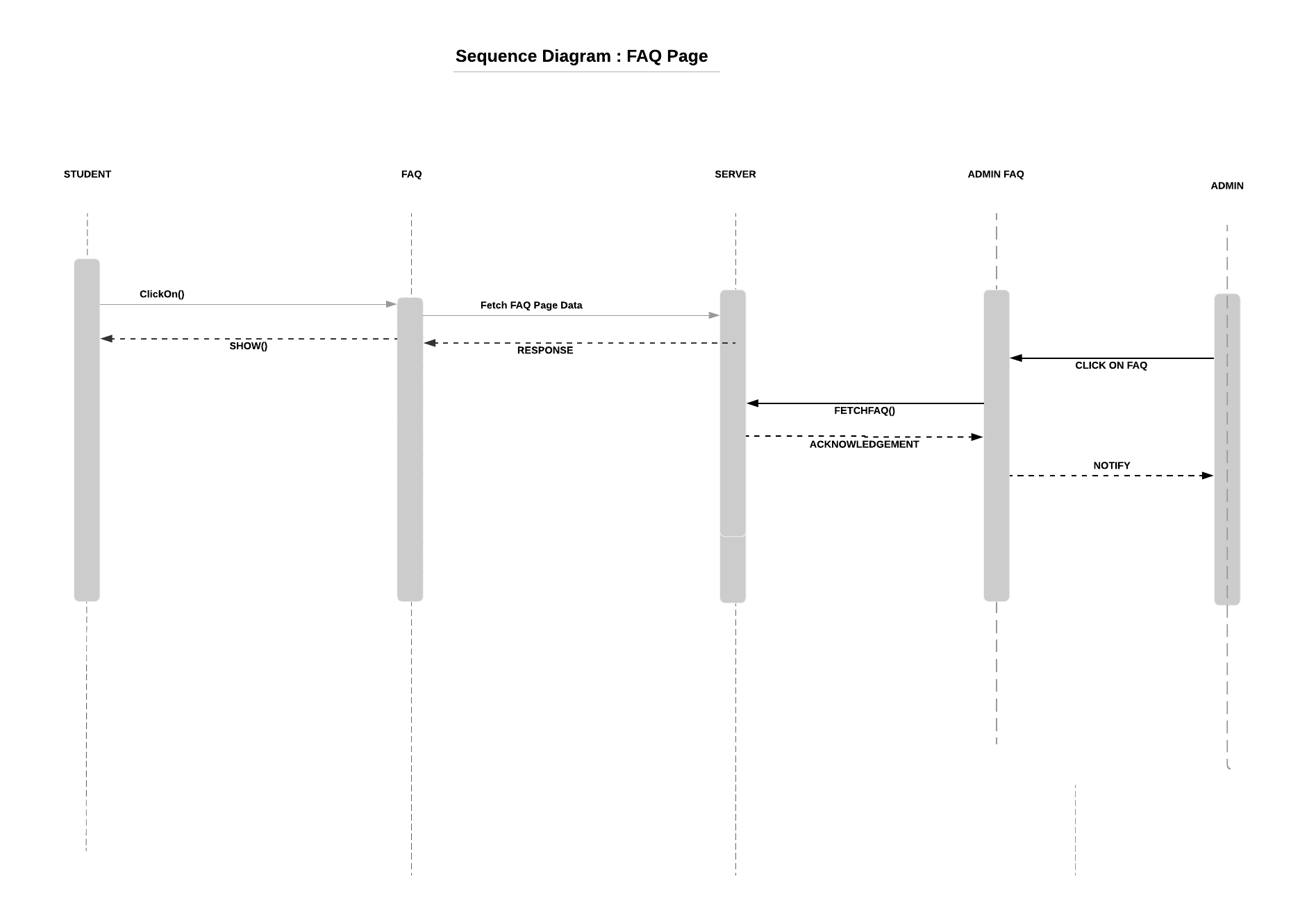
****

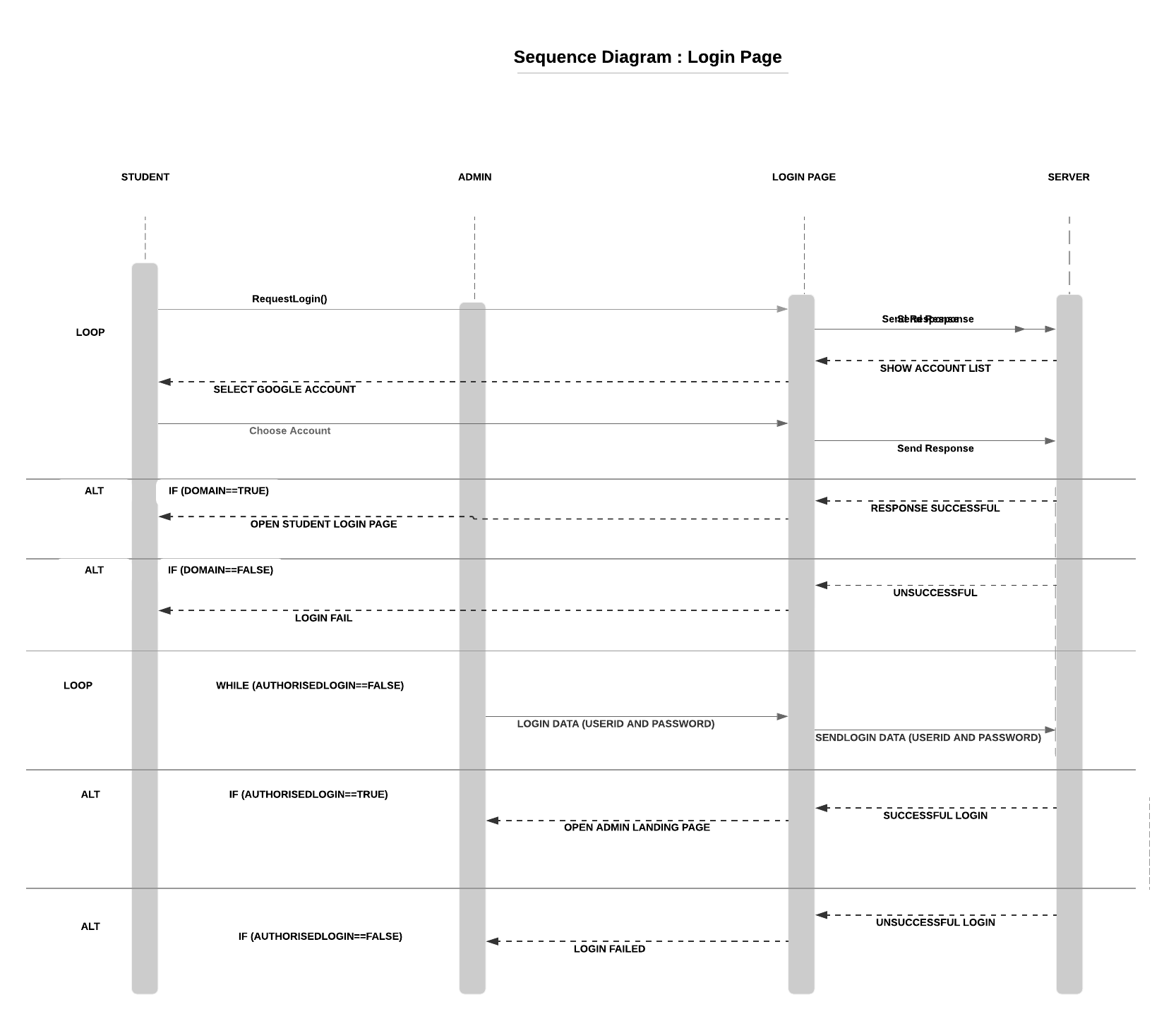
****

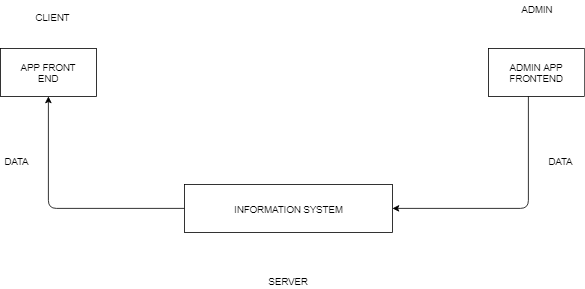
****

****

****

****

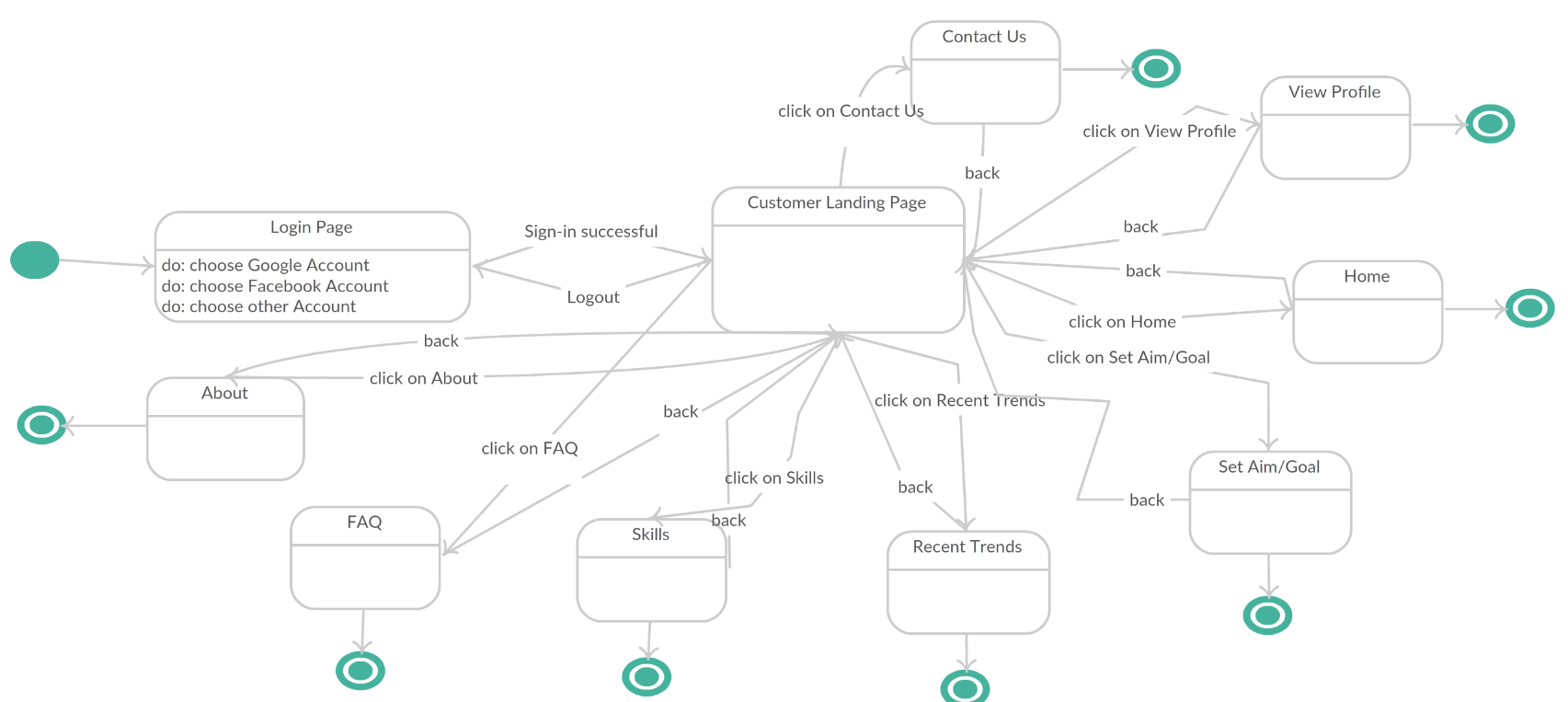
****

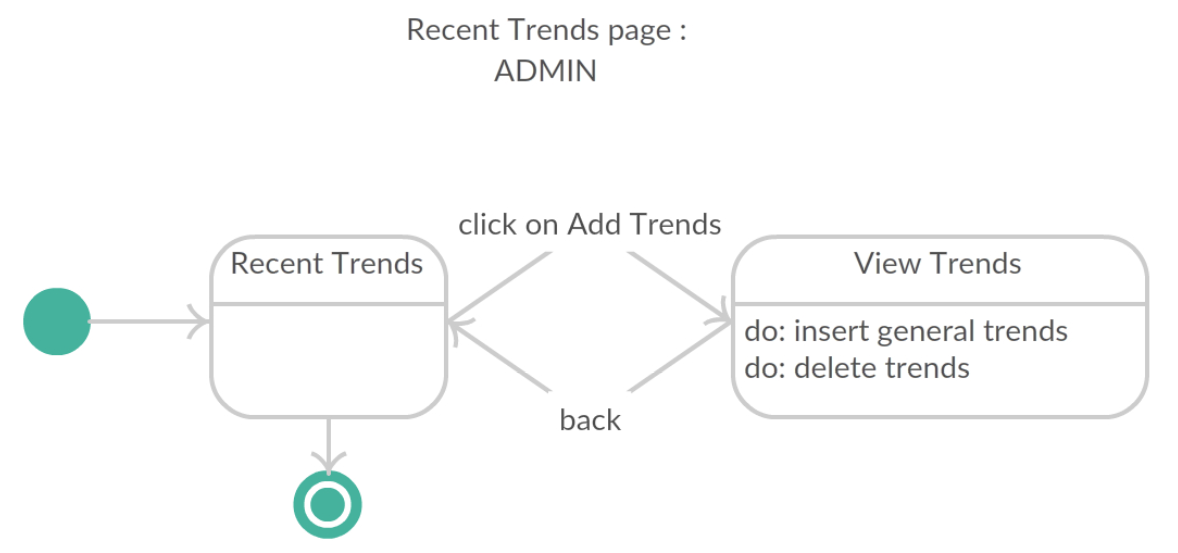
****

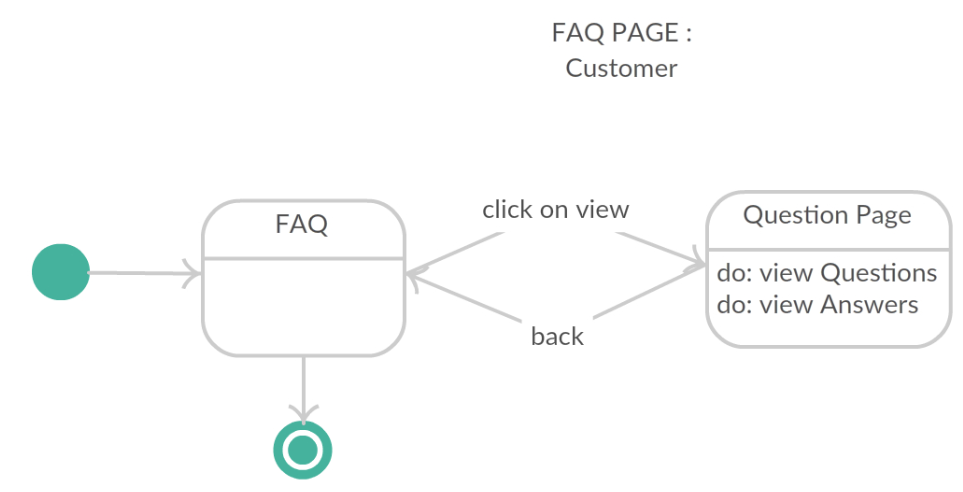


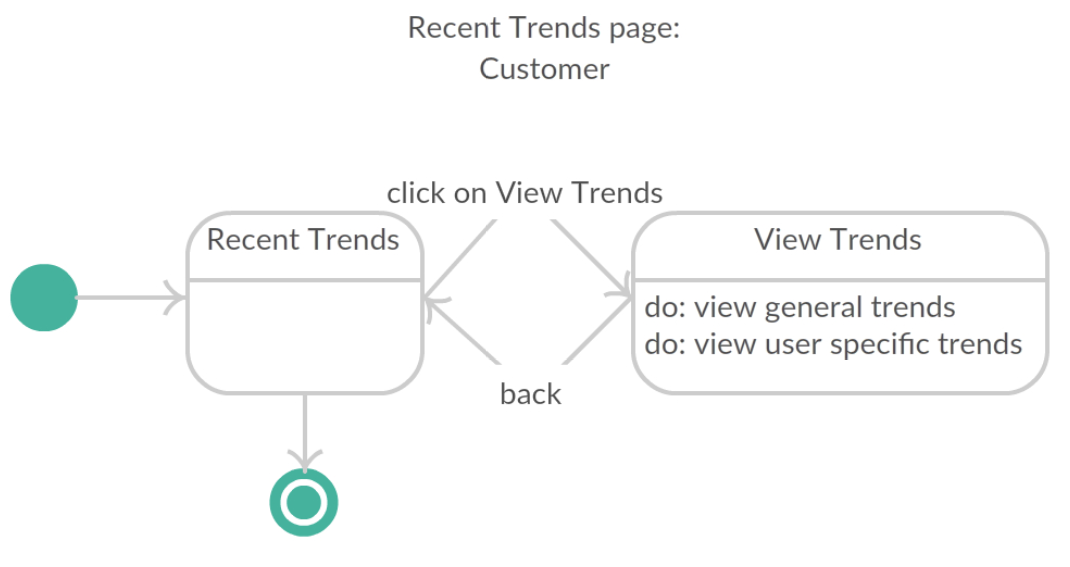
**State Diagrams**

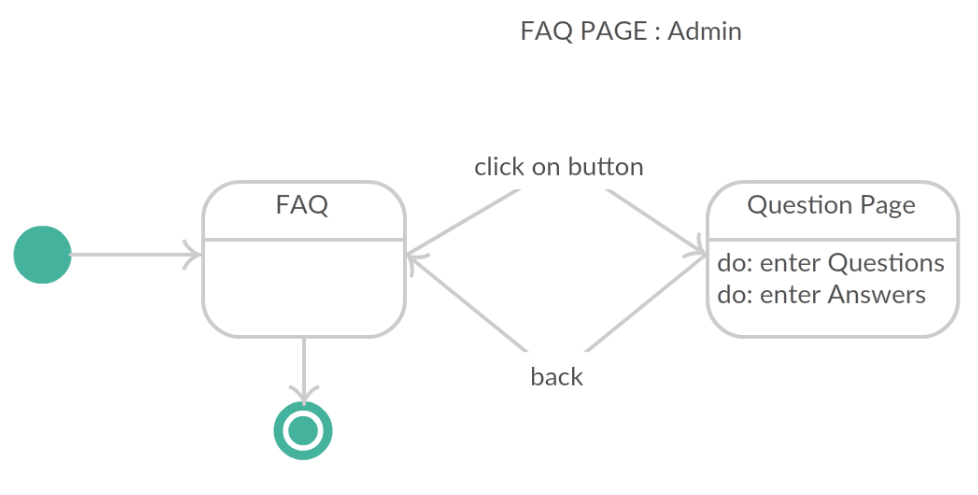
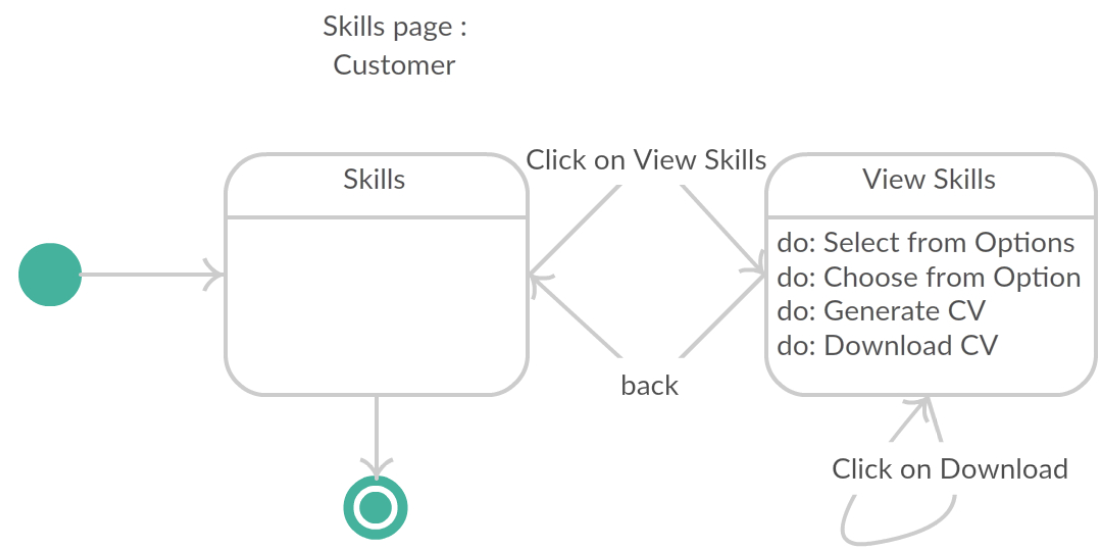
Student Landing Page

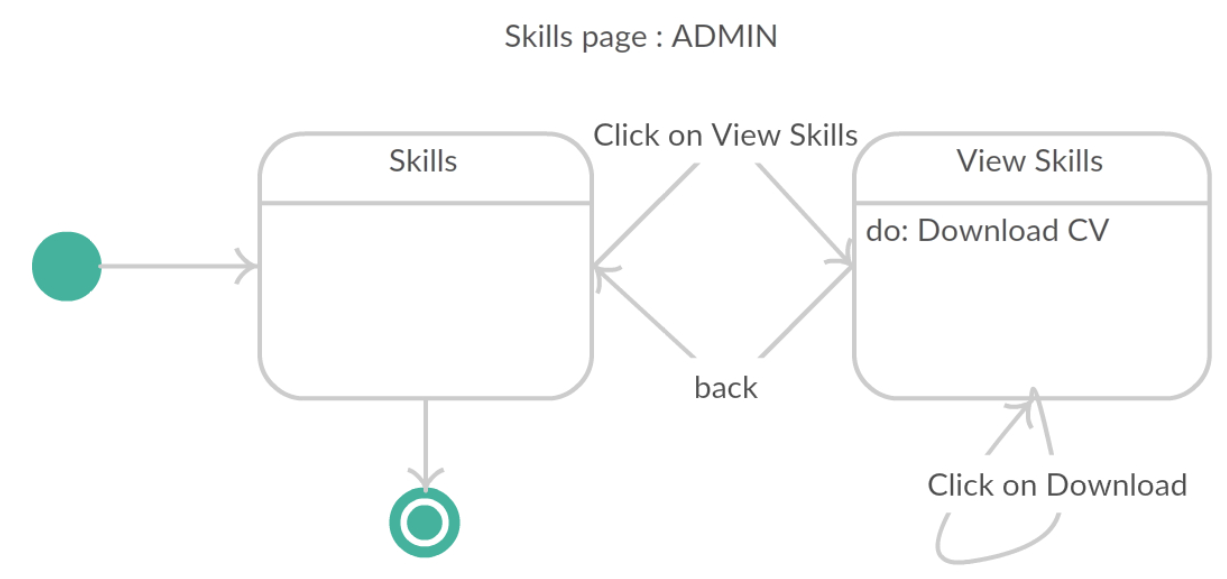


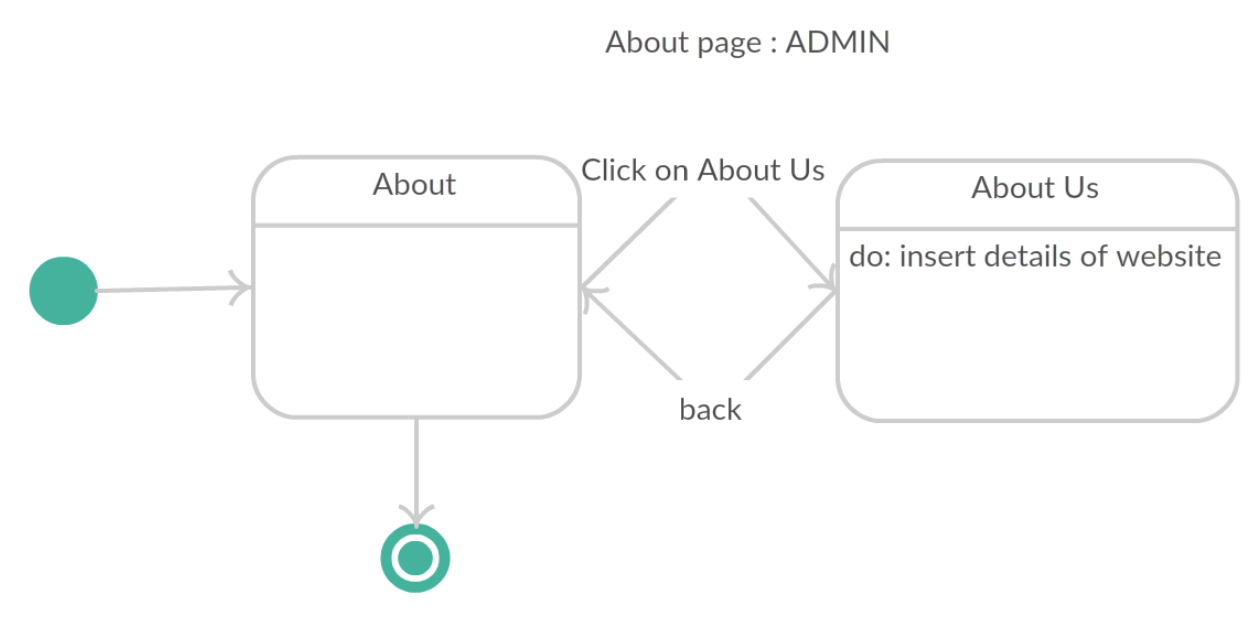


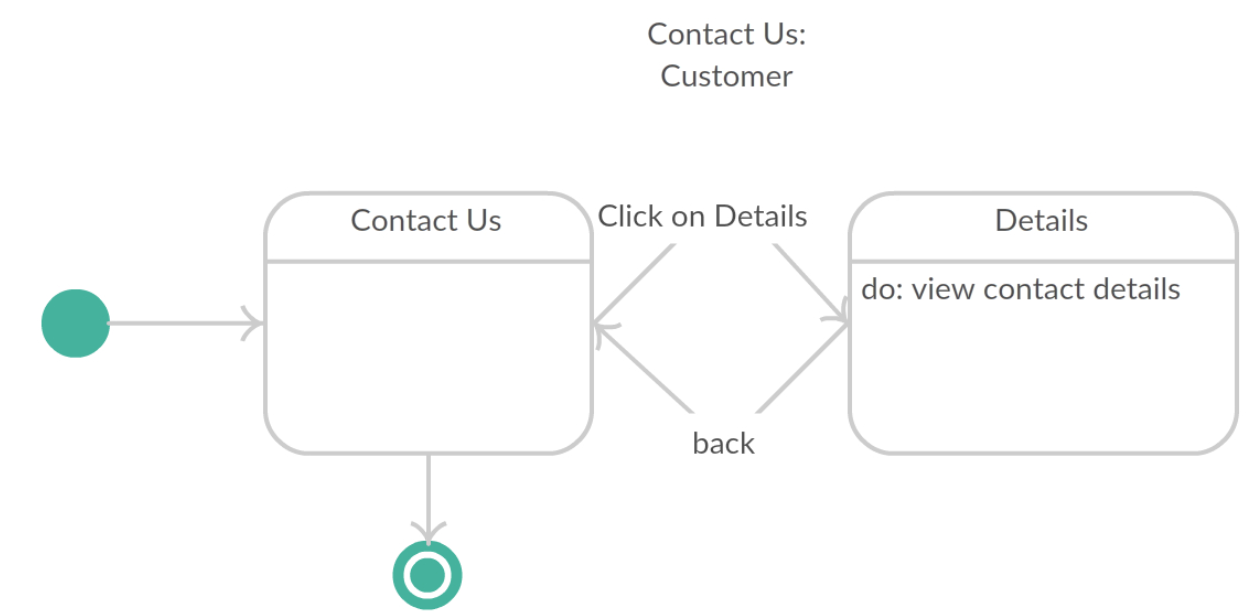


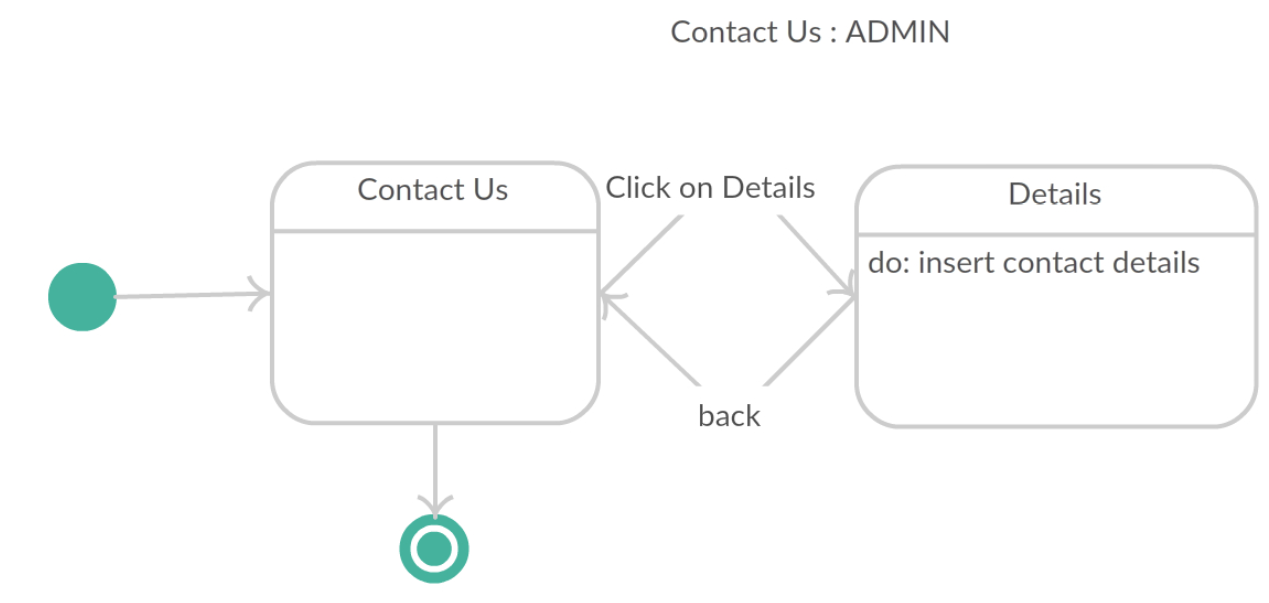


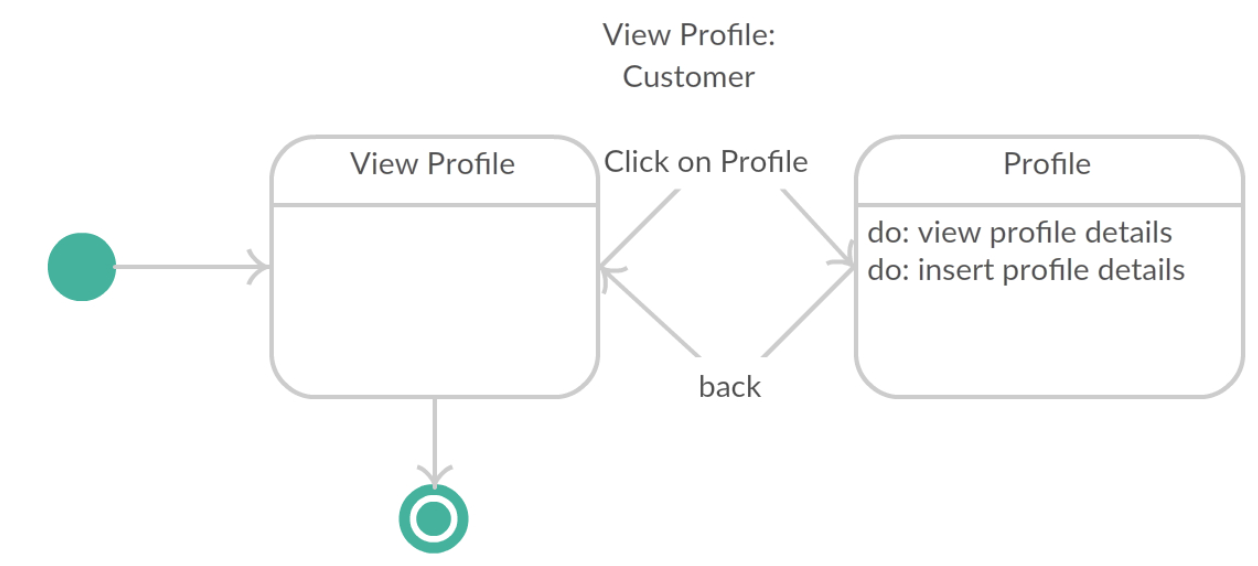


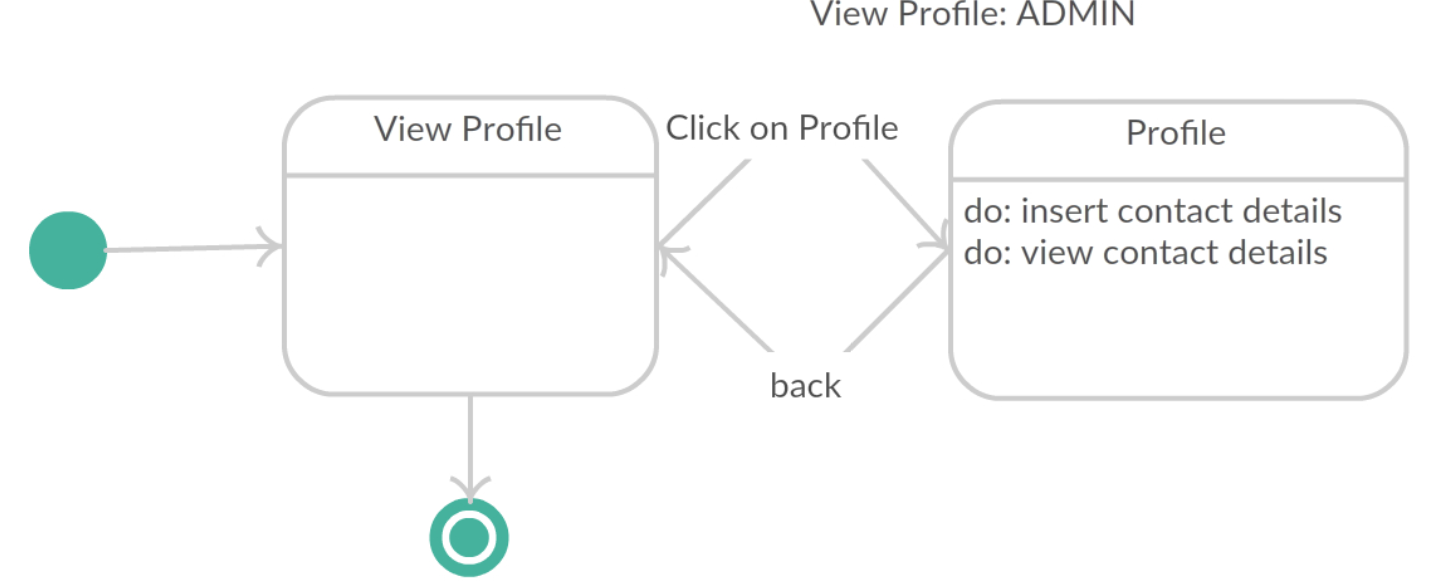


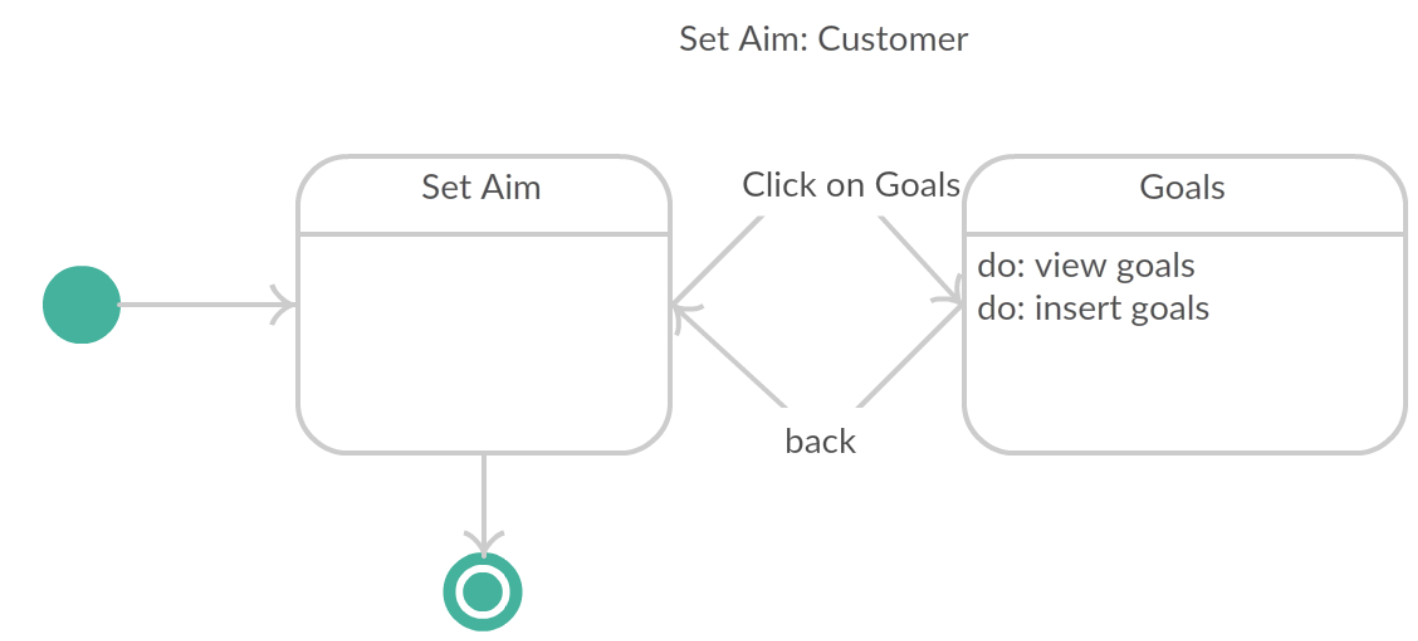


****

****

****

****

****

**3.1 Logical Architecture Description**

Discuss some details(generic) of Logical Architecture

**3.2 X Component (or Class or Function ...)**

Use exactly the template you define in 3.2. If a part of the template is not applicable, then mark it N/A rather than omitting it.

**3.3 Y Component (or Class or Function ...)**

...

**3.n Z Component (or Class or Function ...)**

**4.0 Execution Architecture**

Define the runtime environment, processes, deployment view.

**4.0 Reuse and relationships to other products**

For teams doing enhancement work, reuse is an important issue. Most enhancement work should focus on extending, rather than replacing, the design and product development from earlier semesters. For teams doing new development, reuse can also be an important strategy. In some cases, there is freeware that could be incorporated. In other cases, there are existing modules or classes that could be adapted. Another possibility is the use of special tools that produce open source results and thus permissible under the terms of this course.

This section should include the following subsections as appropriate:

* how reuse is playing a role in your product design
* how reuse is playing a role in your product implementation (and the motivation for changes)
* if you are not reusing material that is available, then give motivation for why it is being thrown out.

**5.0 Design decisions and tradeoffs**

Use this section to motivate any decisions that will help the reader understand

the design that your team is using. This section can also capture good ideas

that were abandoned and the reasons for leaving them out of the design.

**6.0 Pseudocode for components**

**7.0 Appendices (if any)**

**SDS component template**

The template given below suggests a reasonable structure for giving a thorough

description of each component described in Part 3 of the SDS. The specific

information depends in part on the design approach. Your team must adapt this

template to your needs and describe it in section 3.1 of your SDS.

|  |  |
| --- | --- |
| Identification | The unique name for the component and the location of the  component in the system. |
| Type | A module, a subprogram, a data file, a control procedure, a class, etc |
| Purpose | Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements. |
| Function | What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified. |
| Subordinates | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| Dependencies | How the component's function and performance relate to other  components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components. |
| Interfaces | Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here. |
| Resources | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services. |
| Processing | The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| Data | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary. |