

Software Design Document

for

Bravo Six

Version 1.0

Prepared by

Group Name: The Bayesian Conspiracy

|  |  |  |
| --- | --- | --- |
| Alexander Yeang | 11538416 | alexander.yeang@wsu.edu |
| Aidan Mattson | 11578711 | aidan.mattson@wsu.edu |
| Josh Harmon | 11567215 | joshua.harmon@wsu.edu |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
|  |  |
| Date: | 11/8/2019 |
|  |  |
|  |  |

# System Diagrams

## Activity Diagrams

## A screenshot of a cell phone Description automatically generatedHealthcare Benefit Program Application

Fig. 1: Healthcare benefit application activity diagram

The above activity diagram shows the flow of initial user registration and application for a healthcare-type benefit program, demonstrating the entire flow of use cases from user data entry to benefit enumeration, eligibility determination, and form pre-filling (Requirement 3.3.1 of the SRS).

## Unemployment Benefit Program Application

A screenshot of a cell phone

Description automatically generated

Fig. 2: Unemployment benefit application activity diagram

The above activity diagram shows the flow of initial user registration and application for an unemployment-type benefit program, demonstrating the entire flow of use cases from user data entry to benefit enumeration, eligibility determination, and form pre-filling (Requirement 3.3.1 of the SRS). It’s very similar to the previous diagram, but removes the selection of programs step because there’s generally just one program that provides a deterministic benefit.

## A screenshot of a cell phone Description automatically generatedGeneral Welfare Benefit Program Application

Fig. 3: Activity diagram for applying for general welfare benefits

The above activity diagram shows the flow of initial user registration and application for a general welfare benefit type of program, demonstrating the entire flow of use cases from user data entry to benefit enumeration, eligibility determination, and form pre-filling (Requirement 3.3.1 of the SRS). It’s very similar to the first diagram, reinstating the “Select applicable programs” step because somebody might qualify for multiple programs and want to apply for one or more of them.

## Structural Diagram

Fig. 4: Class diagram for the application, omitting potential trivial utility classes

The above class diagram shows the general interaction and structure of the main classes anticipated to be necessary in Bravo Six. The User model fulfills the 3.2.1 requirement to maintain user accounts with some editable user data to avoid repetition when filling out forms. The User has GeneratedForm relationship helps to fulfill Requirement 3.2.4 by providing the user the ability to review recently generated pre-filled forms. The Requirements wrapper around requirements for a program will provide a modular way to define the validation rules for various fields needed for a program application, fulfilling Requirement 3.2.2 to validate user input and provide actionable error messages to the user. The BenefitProgram class, by combining the functionality of its ProgramApplicationForm and Requirements components, can validate the user’s input to determine whether or not they meet the program’s Requirements, fulfilling the 3.2.3 requirement.

The *info* and *requirements* members of the BenefitProgram class facilitate the Requirement 3.1.1 ability to provide summary information about a program on general landing pages or program-specific information pages without requiring the user to go through a whole application process or do their own lengthy research.

## A close up of a map Description automatically generatedBehavioral Diagrams

Fig. 5: Sequence diagram for the benefits application process

Figure 5, above, depicts the general data flow and process of the benefits application process from a system perspective. Importantly, the Frontend providing the user with Validation Errors means that client-side validation will be employed (augmented by server-side validation, of course) to give users actionable feedback as soon as possible, meeting the 3.2.2 and 4.1 Requirements.

The system will begin in the upper half of the diagram by combining the user’s profile information with information entered into a program-specific form, and will compare that combined information against program requirements on the server side. If the user is eligible for the program, that will be relayed back to them, at which point they can opt to generate pre-filled forms for the relevant program, kicking off the lower half of the sequence diagram, which will actually fetch the forms (or instructions) needed to officially apply and will fill the user’s information in as completely as is feasible, following pre-set instructions in the BenefitProgram’s FilledFormGenerator. Once generated, the Backend will store it on some storage medium and return a link to the user to access the generated document for a limited amount of time.

No other behaviors are expected to be complicated enough to model, as they are simple CRUD operations. For example, an index page of benefit programs would simply fetch all programs from the database and return some information for each of them without any meaningful user interaction or particularly noteworthy system behavior. Likewise, the user registration process will be like traditional web registration forms, asking for simple information, validating it, storing it in a database, and then carrying on, and as such we do not believe it is worth modeling.