Software Detailed Design

Data Design

The data design defines each of the objects within the application, divided into two categories:

* Pages – primary concerns are displaying information and transitioning between pages
* Non-page objects – primary concerns are containing and retrieving information

Pages

Pages are the individual “units” of the user interface. They display information, accept user input, and govern the flow of the application, from page to page.

LandingPage

Responsibilities:

* Title page
* Displays briefly

Attributes:

* String title – title of application

GpsPage

Responsibilities:

* Get phone’s GPS
* Find plants nearby
* Display those plants
* Send user to plant profile
* Query database for nearby plants
* Allow user to go to RefinePage

Collaborators:

* PlantDataObject
* RefinePage
* PlantProfilePage

Attributes:

* Gps gps – GPS coordinates of user’s phone

Methods:

* displayRefineButton() – displays the refine button, which calls loadRefinePage()
* loadRefinePage() – loads RefinePage when refine button is clicked
* getGPS() – get phone’s GPS
* loadPage(String pageName) – loads the page that’s passed in
* displayPlantList(gpsData gps) – display list of plants based on location

RefinePage

Responsibilities:

* Allow user to choose between options
* Display options
* Get next options
* Display list of plants so far (in carousel)
* Allow users to select plant profile
* Load plant profile
* Query database for plants based on options

Collaborators:

* PlantDataObject
* PlantProfilePage

Attributes:

* List<String> selectedQualities – list of selected qualities to filter on

Methods:

* getNextOptions() – return next set of options to choose from
* loadPage(String pageName) – loads the page that’s passed in

DirectLookupPage

Responsibilities:

* Allow user to directly lookup a plant based on knowledge of genus and species

Collaborators:

* PlantDataObject
* PlantProfilePage

Attributes:

* Plant selectedPlant(String genus, String species)

Methods:

* displayGenus() – call getGenuses(), iterate through list of remaining plants, and display
* displaySpecies() – call getSpecies() and display the plant that’s returned
* loadPage(String pageName) – loads the page that’s passed in

PlantProfilePage

Responsibilities:

* displays information about a specific plant

Collaborators:

* PlantDataObject

Attributes:

* PlantDataObject currentPlant – the plant to display information about

Methods:

* loadPlantDataObject(String genus, String species) – load data for specific plant
* loadPage(String pageName) – loads the page that’s passed in

Non-Page Objects

PlantDataObject

Responsibilities:

* Hold data about a single plant

Collaborators:

* DataAccessObject

Attributes:

* String commonName – common name of plant
* String scientificName – scientific name of plant
* String imageUrl – link to image of plant
* String location – some type of location for the plant (either county or long/lat)

DataAccessObject

Responsibilities:

* Directly access database

Collaborators:

* Al other non-page objects

Attributes:

* Int displayThreshold – number of unique species below which results should be displayed
* GpsData gps – user’s GPS coordinates (from phone)
* Date date – current date
* List<String> qualityList – list of selected plant qualities
* Int count – number of returned results

Methods:

* getCount() – get count of plants in plantList

DirectLookupObject

Responsibilities:

* Contain data inputted from DirectLookupPage

Collaborators:

* DataAccessObject
* PlantDataObject

Attributes:

* String selectedGenus – genus that was selected
* String selectedSpecies – species that was selected

Methods:

* getSpecies(String genus) – returns full list of species within genus name that is passed in

Architecture Design

Subsystem Decomposition

**Database** – The data storage mechanism in our case will be a database of plant species throughout the united states. It will also use geo-location to determine the amount of plants displayed on the device. It will be maintained by other institutions such as herbariums, colleges, and other scientific organizations.

**Determining API** – This system will be a third-party application which communicates with different databases and or one whole database that will retrieve information from those databases. This will have the access point for the data via a defined API.

**Application** – The application will be developed and maintained by our team and will provide the logic which will govern the organization and presentation of data as well as the options for data interaction which will be available to the user.

Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that will be imported into apps.

* **Modules:** Declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities. It will be stored in a root module typically named AppModule.
* **Components**: Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment.
* **Templates**: Combines HTML with Angular markup that can modify HTML elements before they are displayed.
* **Directives**: provide program logic, and binding markup connects your application data and the DOM. There are two types of data binding:
  + Event binding lets your app respond to user input in the target environment by updating your application data.
  + Property binding lets you interpolate values that are computed from your application data into the HTML.
* **Services**: Share across components
* **Dependency Injection**: Lets you keep your component classes lean and efficient.
* **Routing**: Defines a navigation path among the different application states and view hierarchies in your app.

**User Interface** – Users will use touch screens to interact with our system on physical mobile devices. The user interface will be rendered via an Android or iOS operating system and API/libraries, depending upon the device. This system will be designed and maintained by a third party.

Hardware/Software Mapping

**Database** – The “database” in our case will actually consist of a physical and/or virtual databases which will be maintained by a third-party institution. Access with the databases will be abstracted (from the developer and the application perspective), and we will not be responsible for maintaining the database(s).

**Application** – The application code will be deployed to the Android and iOS application stores, and users will be able to access the code from there and “deploy” the software to their own physical devices by downloading it. Once downloaded, the application will reside on the user’s physical device.

**User Interface** – The user interface will reside entirely on the user’s device, which will be maintained by the user and the third parties that manufactured it and provided the operation system.

Interface Design

Some navigation will be handled via a Navigation XAML page, which will be function as a kind of “frame” for the other pages. This Navigation frame will include a back button on every page and will allow for the loading of adjacent page objects in the control flow graph.

The individual pages were briefly described above, but they are as follows:

* LandingPage – Also known as the “title” screen, fades slowly into visibility and displays title screen.
  + Can be loaded from:
    - No page, application load only
  + Can load:
    - DirectLookupPage
    - RefinePage
* GpsPage – Displays results based on physical proximity to user
  + Can be loaded from:
    - PlantProfilePage
  + Allows for loading of:
    - RefinePage
    - PlantProfilePage
* RefinePage – Also known as the “key,” allows user to select between a variety of plant qualities, filtering and displaying increasingly fewer plants
  + Can be loaded from:
    - LandingPage
    - GpsPage
    - PlantProfilePage
  + Allows for loading of:
    - GpsPage
    - PlantProfilePage
* DirectLookupPage – Allows user to select a genus, and/or species to quickly jump to a particular plant profile
  + Can be loaded from:
    - LandingPage
  + Allows for loading of:
    - PlantProfilePage
* PlantProfilePage – Displays information about a specific plant
  + Can be loaded from:
    - GpsPage
    - RefinePage
    - DirectLookupPage
    - PlantProfilePage
  + Allows for loading of:
    - All pages