AndroidMonke Productivity

Dr. Sepideh Ghanavati

6 March 2022

COS420 - Software Engineering

Architectural Design of NO Monke-ing Around!

The architectural model of NO Monke-ing Around! (NMA!) will be that of Model-View-Controller (MVC) blended with some Cloud computing. The model will be directly shown to the user via the view where the user will be able to access different tabs of the application in order to access certain features of their application. AndroidMonke Productivity (henceforth referred to as “we”) have decided that MVC is the strongest architectural design for our application due to MVC focusing specifically on the user-facing interface, the reliance on our application’s built in features in order to manipulate the viewing and behavior of the application, and we do not want the user’s being able to access any data directly.

To begin, MVC allows the user to directly manipulate the view and apply state changes to the model. The view manipulation in NMA! will primarily be concerned with the switching of the user’s “page,” meaning that the user will be able to swap between the settings, leaderboard, main, calendar, and tree collection pages of our application through requests to manipulate the view. The view, in return, will directly render the model necessary for each page to be displayed. For example, if the user were on the main page and switched to the leaderboard page, the tree from the main page must be animated to shift off the screen as two animated wooden signs scroll in from the left hand side of the screen, taking the place of the tree that was once there. The user also has the ability to manipulate the model of the application. To manipulate this model, the user may simply request to grow a tree. The action of growing a tree will in turn demand a state

change from the view to one where the sapling is displayed, along with messages about what the user should be focusing on, and the user's focus timer.

Our application will directly be able to handle all changes to view and model as well, meaning that MVC is single handedly the best architectural design for NMA!. The view shifting will be done natively with XML as the primary language behind it. The user will be able to click or drag between views in order to display whatever the user would like. We already have a mock-up of the view changing, where swiping between the different pages causes the prior to slide out from the relevant side and the desired page to be displayed. The model manipulation will be handled through XML as well. When the user requests to plant a tree, the screen will change and a new model will be presented to the user. The user, in this instance, will have requested a state change from the Model, which in turn will send a change notification to the View, which will then trigger a ussr event that changes the screen.

The inaccessibility of data to the user via MVC is pivotal to NMA!'s success as well. MVC does not allow the user to interface with data directly, rather forcing these interactions to be done through changes to the view and model. The data, via the view and model, will be able to shift without the presentation shifting, which is pivotal for scoreboards and such to activate. For example, while the user is growing a tree, they are accumulating focus hours, however the focus hours would be displayed on the activity tracker tab, rather than the main page where the user's tree is. Thus, the viewing of the data must be independent of the user's view. The data management of NMA! will be handled via the cloud computing architectural design.

The cloud computing will allow for a database to be used to handle the users information. The information we will store will be due dates, google account logins, usernames and passwords of the users, the last ten trees of the user, the amount of time the user has spent

focusing in the last two weeks, and whether the user would like NMA! to plant trees for them or not. The form of cloud computing used by NMA! will be IaaS (Infrastructure as a Service) where we will be using cloud servers, storage, and system management, accessible via the application's connection to the internet. The general users will not have access to the cloud computing, it will all be handled on the backend.

In conclusion, No Monke-ing Around! will use a mix of primarily Model-View-Controller architectural design and secondarily Cloud Computing architectural design. The MVC will handle the frontend of the application, through the presentation and animations, where the cloud computing will handle the backend of the application, through the data storage and management. The user will not have access to the Cloud Computing architecture within the application.