print(d)

3D Printing made easy

Adam Tremonte, Andrew Thomas, Emily Huynh, and Patrick Dixon

CHANGE HISTORY

VERSION	SUMMARY	AUTHOR	DATE
0.1	Initial Draft	All	02/05/2017
0.2	Added project description and requirements	All	02/09/2017
0.3	Diagrams	All	02/23/2017
0.4	Activity Diagrams	All	02/28/2017
1.0	First Official version (Requirements)	All	03/02/2017
1.1	Adding sequence diagrams with descriptions, adding descriptions to other diagrams, and adding detailed class diagram.	Adam, Andrew, and Patrick	03/29/2017
1.2	Adding more class diagrams, adding descriptions	All	03/30/2017
1.3	Added more sequence diagram descriptions	Emily	04/02/2017
1.4	Final	All	04/28/2017

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Document Purpose

The purpose of this document is to define the formal requirements of the Print(d) application. It outlines the functional and nonfunctional requirements along with the intended features of the application.

1.2 Scope

This project will consist of an Android application that integrates Thingiverse, OctoPrint, and a slicing server with a user-friendly interface. Modules include the slicing server, Thingiverse API, OctoPrint API, and printer configurations.
In addition to the application, a backend server will be created to allow users to convert digital models to a file format their printer can understand.

1.3 Definitions

- **3D Printing -** A manufacturing process known as Additive Manufacturing, where parts are fabricated through building layer on layer.
- **3D Printer -** A machine that can extrude material in successive layers to produce a part from a digital model.
- **API (Application Program Interface) -** a publicly available interface that provides developers with programmatic access to a proprietary software application.
- **Bed -** The bottom plane onto which the 3D printer extrudes filament.
- **Extruder -** The “hot end” of the printer. A nozzle that reaches high heat to melt filament and position it to form a model.
- **FDM (Fused Deposition Modeling) -** A type of Additive Manufacturing commonly used for many commercially available 3D printers.
- **Filament -** The material used to print with. Usually a roll of 1.75mm PLA plastic.
- **Gcode** - The file type used by 3D printers.
- **Model** - The part that is to be printed.
- **OctoPrint** - An open source, internet connected controller for most models of 3D printers. Typically run on a Raspberry Pi or other hobbyist cheap computer.

- **Raspberry Pi -** A small, cost effective hobbyist computer usually used for standalone projects, such as controlling an OctoPrint server.
- **Slicing Software** - A program designed to take a STL file model and convert it into Gcode that a 3D printer can read.
- **Thingiverse** - A website, created by MakerBot Industries, designed to be a place for people to upload and share models designed to be 3D printed.

2. PROJECT DESCRIPTION

2.1 Product Purpose

We've noticed how difficult it can be to get into 3D printing, so we are designing a better, more user-friendly approach for consumers to be able to learn 3D printing.

Print(d) will allow users to configure their printer and print custom models without ever having to interact with gritty printer details or modeling and slicing software. The software interface will help walk users through previously tedious tasks such as bed calibration and model slicing, all within one centralized space.

2.2 Product Features

2.2.1 Android Application

We will develop a mobile application that connects all of the core functionality. The application will be connected to the user's Print(d) and Thingiverse accounts. Once logged in, the user will be able to select 3D models from Thingiverse and send it a local OctoPrint server for 3D printing.

2.2.2 Thingiverse API

Our mobile application will connect the user's Thingiverse account with their Print(d) account. The user will then be able to pull Thingiverse collections to select for printing.

2.2.3 OctoPrint Server

After the model has been selected, the user will use OctoPrint to interface with the desired 3D printer. We will provide basic instructions for the user to set up an OctoPrint server (if it is not already set up) on a Raspberry Pi that is connected to the printer. Once connected, OctoPrint will slice and instruct the printer to print the selected part.

2.3 Constraints

- Must be on same wifi network as the Raspberry Pi.
- Constrained to just Thingiverse, can't use similar sites.

2.4 Assumptions & Dependencies

- Thingiverse and OctoPrint APIs
- The user's devices are wifi enabled and are connected to the internet
- Heroku online

3. FUNCTIONAL REQUIREMENTS

3.1 User Account Management

DESCRIPTION	PRIORITY
Account Creation	High
Login	High
Logout	Low
Delete Account	Low

3.2 Setting up printer profile within app

DESCRIPTION	PRIORITY
Build Volume	High
Make/Model	High
Heated Bed checkbox	High

3.3 Pulling collections from Thingiverse

DESCRIPTION	PRIORITY
Thingiverse User Authentication	High
Collection Tracking	High
Ability to retrieve a model (.stl) file from a user's collection	High

3.4 Remote server (Heroku)

DESCRIPTION	PRIORITY
Heroku backend server will be able to receive a .stl file and utilize the Cura Engine to slice the file into a .gcode file	Medium
Server will be able to authenticate users and manage account creation	Medium

3.5 Pushing to OctoPrint Server

DESCRIPTION	PRIORITY
Connect mobile app to OctoPrint API	High
Send sliced model to OctoPrint server	High

3.6 Filament Tracking

DESCRIPTION	PRIORITY
Total filament input and storage	Low
Track filament per job	Low

3.7 Alignment / Bed Level

DESCRIPTION	PRIORITY
Direct control of 3D Printer over OctoPrint to move extruder head to necessary positions	Medium
Companion interface to walk users through the process	Medium

3.8 Error Handling

DESCRIPTION	PRIORITY
Separate error messages for each step	Low

4. NON-FUNCTIONAL REQUIREMENTS

4.1 Hardware Requirements

- The user must have a 3D printer
- The user must have an OctoPrint server connected to their 3D printer
- OctoPrint must be connected to the internet
- The user must have an Android device

4.2 Location Requirements

- The user must be connected to the same wifi network as the desired OctoPrint server

4.3 Service Requirements

- The user must have a Thingiverse account

4.4 Usability Requirements

- The Print(d) application must be easy to use for users that have little to no experience with 3D printing.

4.5 API Requirements

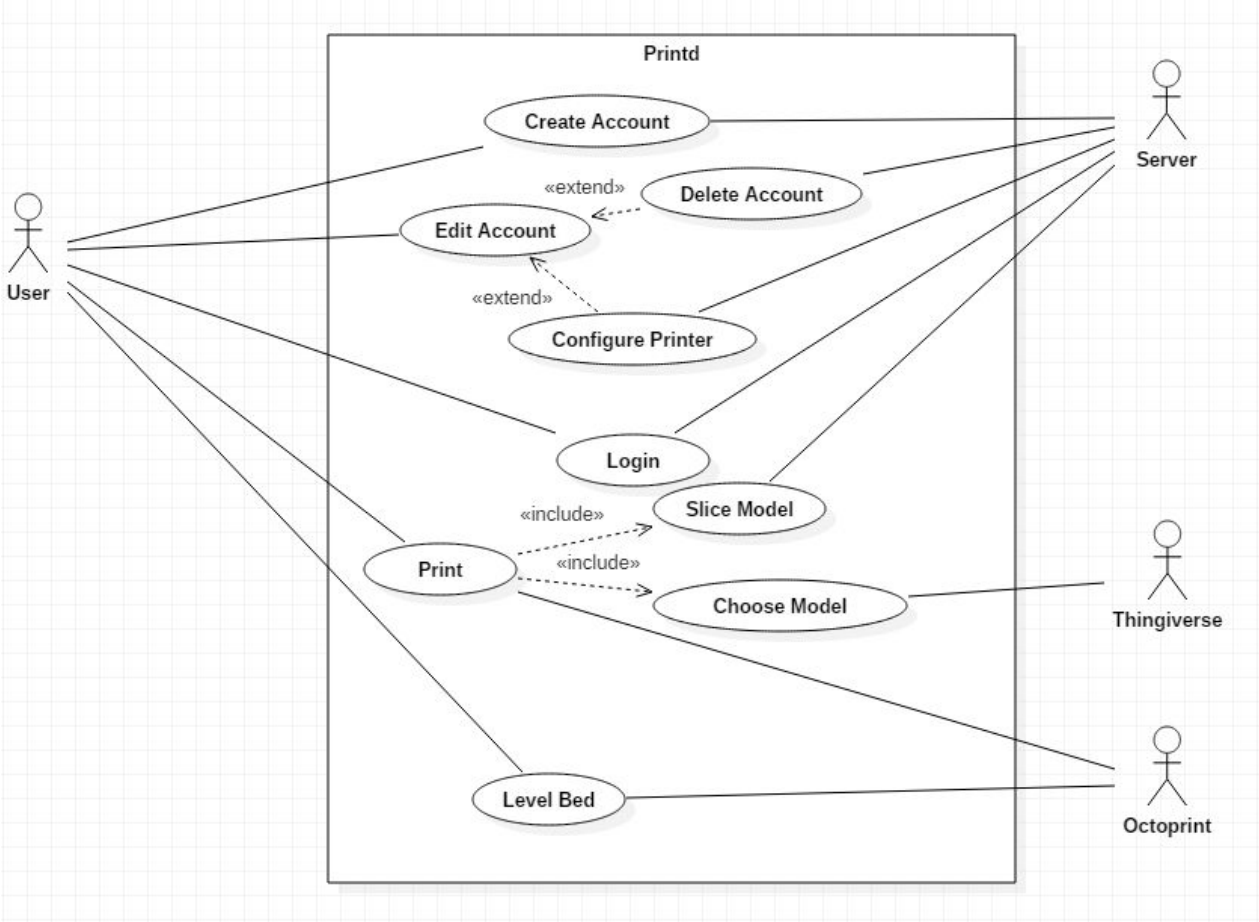
- Thingiverse API open and available
    - Allows users to view their collections
- OctoPrint API allows for direct printer control and pushing Gcode files

4.6 Backend Server Requirements

- The server will be able to communicate through an API to the Android application
- The server will be able to store user data

5. UML DIAGRAMS

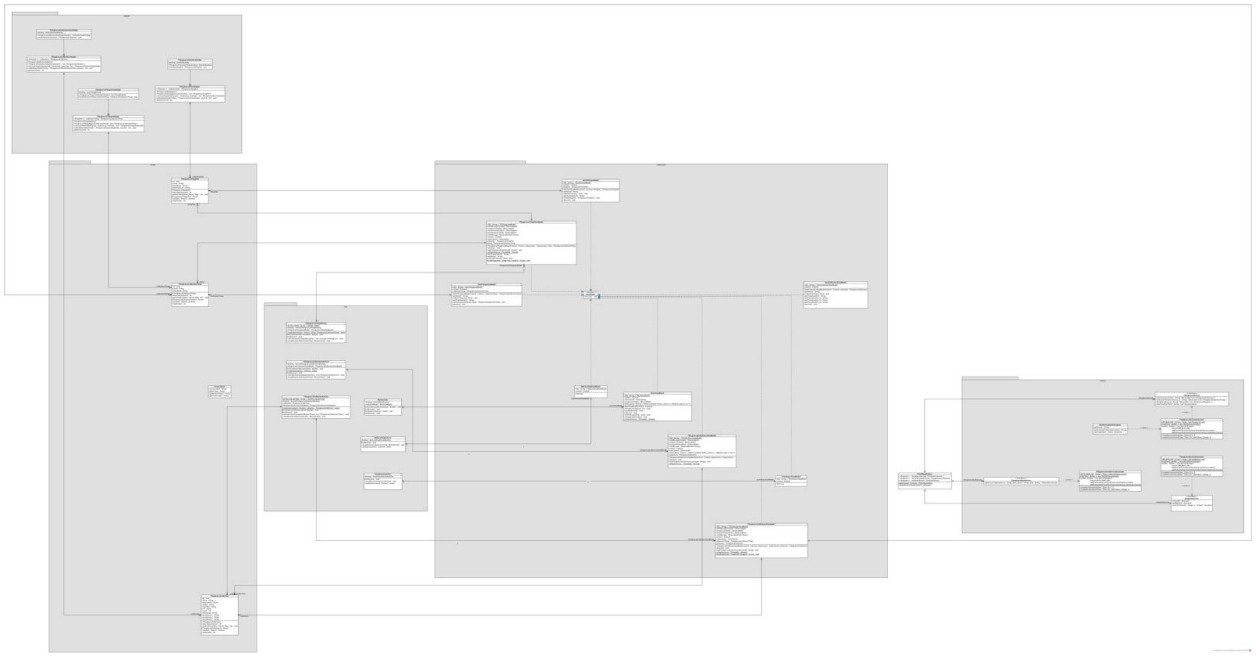
5.1 Use Case Diagram



This is a basic overview of how printd will work. The user will just communicate with the app for creating an account, editing an account, printing, and bed leveling. Printd then talks to the appropriate service to be able to satisfy that request. All account management tasks will be handled by the Heroku server. The bed leveling and slicing will be handled by the OctoPrint API. The data loaded from Thingiverse will be handled by the Thingiverse API.

5.2 Class Diagram

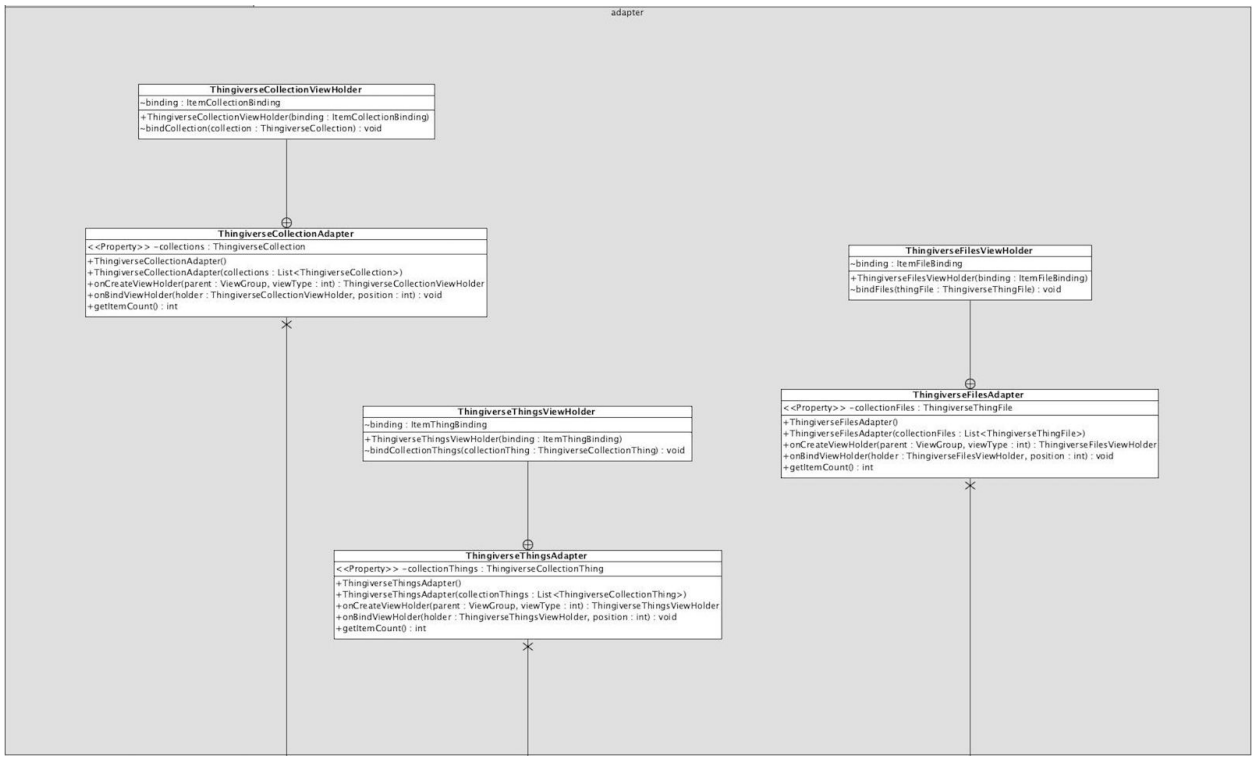
5.2.1 Main Class Diagram



The class diagram can be divided into six main parts: five namespaces, and the application class. The MVVM framework being used to develop Print(d) has led to a multitude of classes, but these classes allow for clean code that is succinct and well organized. The five namespaces, starting from the top left and ending on the bottom right, are as follows: Adapter, Model, View, ViewModel, and Service. Each namespace is enlarged in the following subsections.

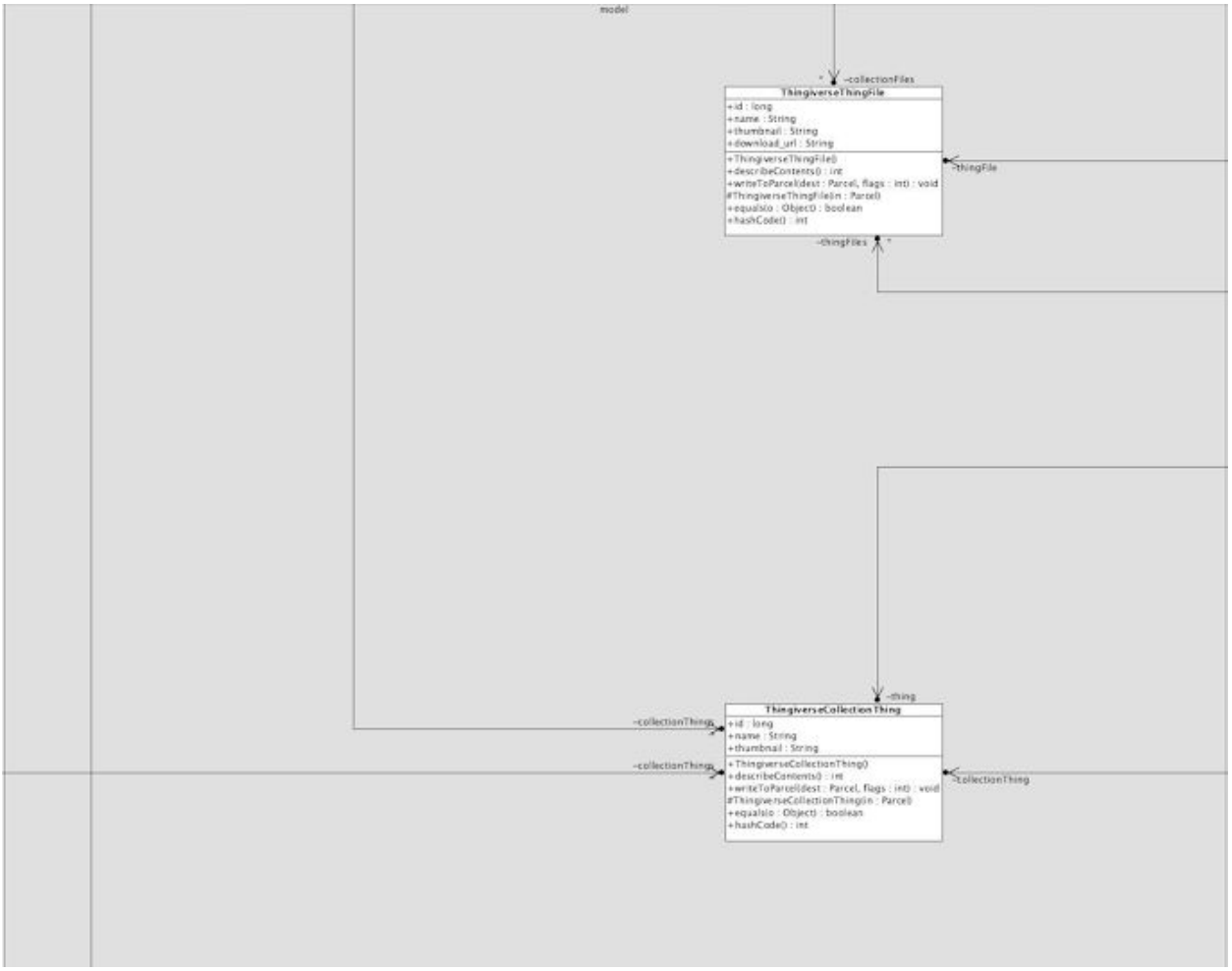
To view the class diagram in full size, please visit this link to our website: https://print-d.github.io/images/class_diagram.png

5.2.1.1 Adapter Namespace

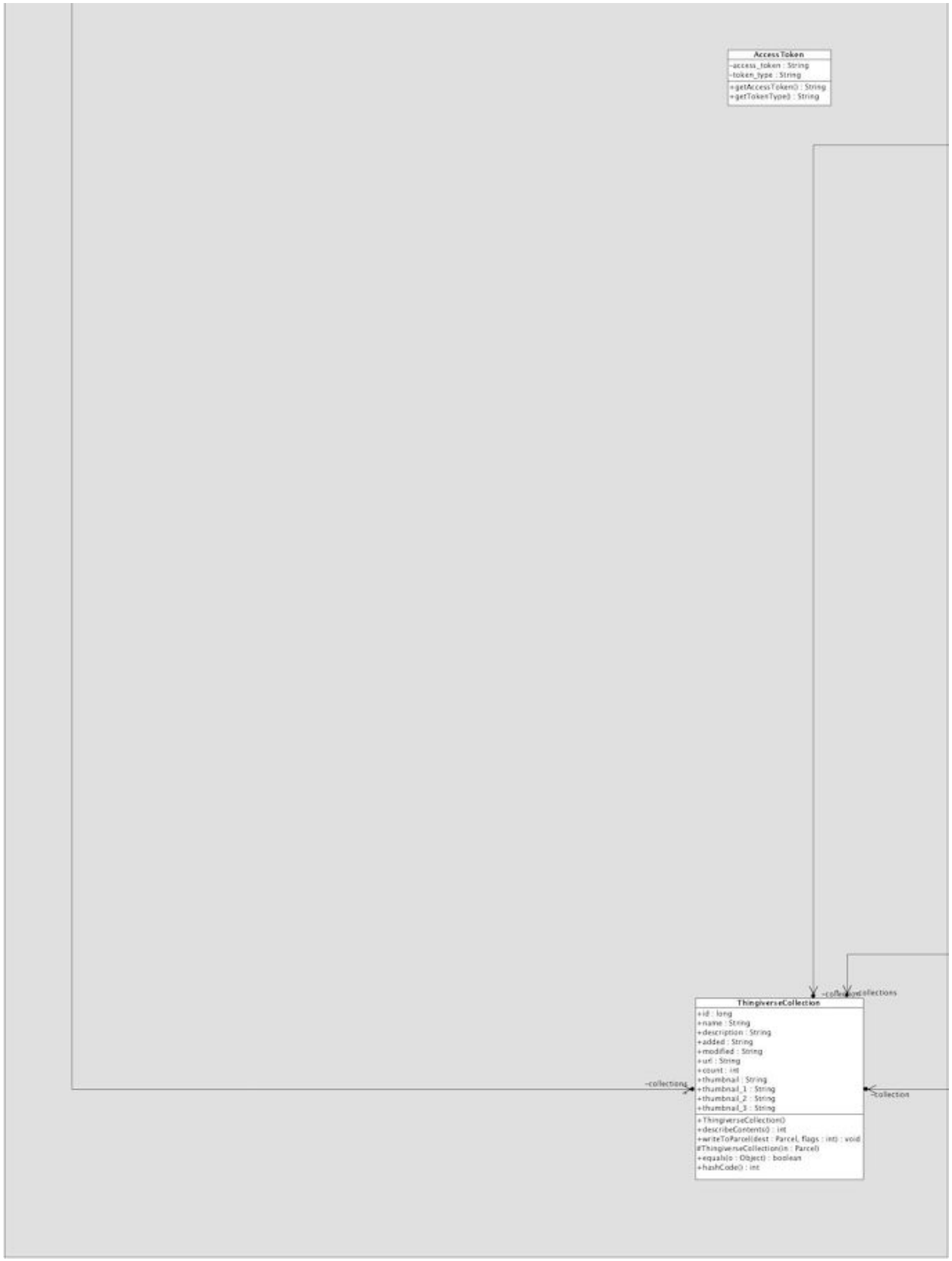


The Adapter namespace is utilized when RecyclerViews are needed within an Activity. A pair of classes, namely a ViewHolder and an Adapter, are used in conjunction to ensure each RecyclerView is appropriately filled with correct information and uses the correct viewmodels.

5.2.1.2 Model Namespace

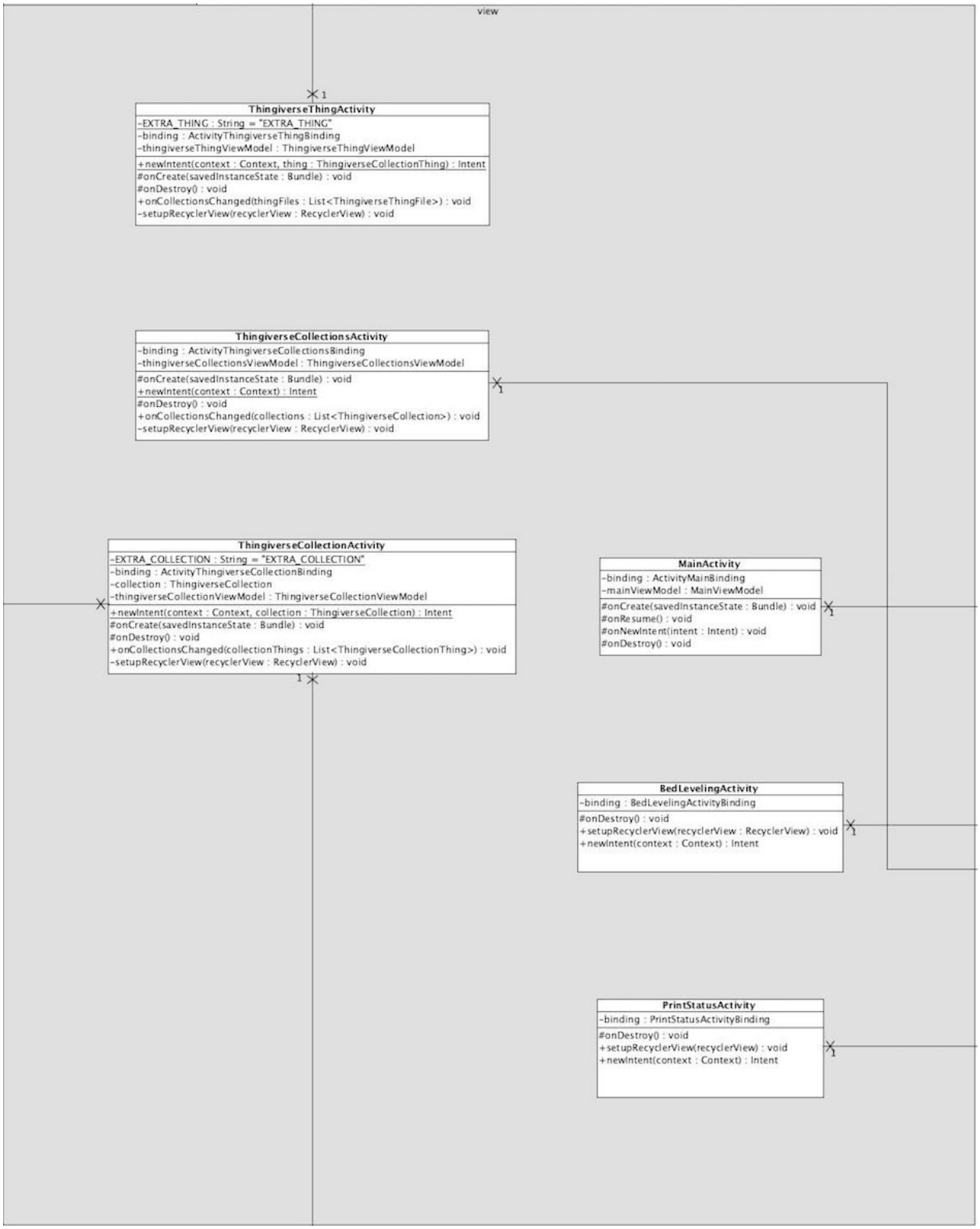


(Model namespace continuation and description are on the next page)


AccessToken
-access_token : String
-token_type : String
+getAccessToken() : String
+getTokenType() : String
-collectedCollections
ThingiverseCollection
+id : long
+name : String
+description : String
+added : String
+modified : String
+url : String
+count : int
+thumbnail : String
+thumbnail_1 : String
+thumbnail_2 : String
+thumbnail_3 : String
+ThingiverseCollection()
+describeContents() : int
+writeToParcel(dest : Parcel, flags : int) : void
#ThingiverseCollection(in : Parcel)
+equals(o : Object) : boolean
+hashCode() : int
-collections
-collection

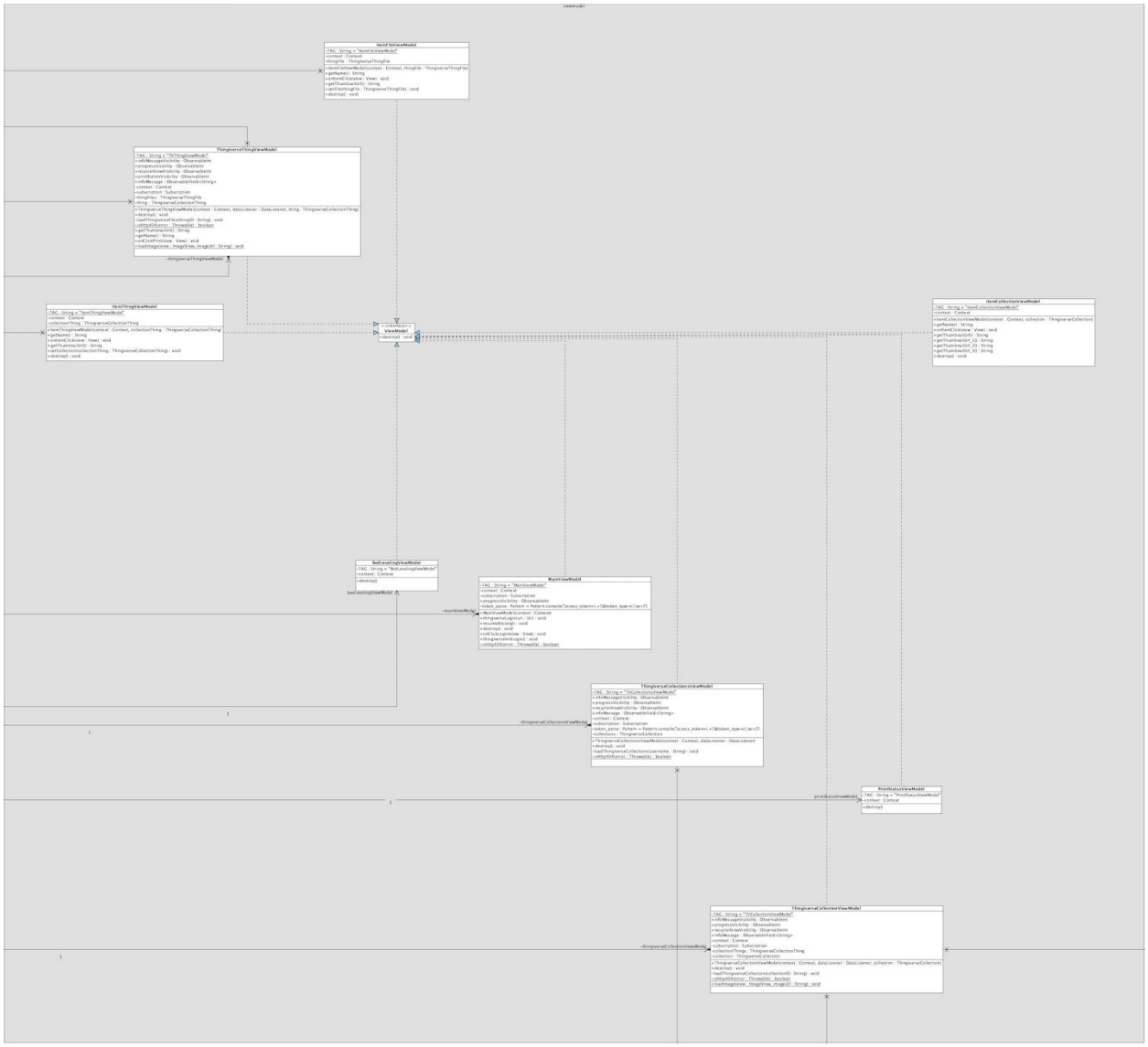
The Model namespace is used for data classes to store data from APIs or within the application in a logical, organized format. Viewmodels interact with data through these Model classes, and Services return data organized into these Model structures.

5.2.1.3 View Namespace



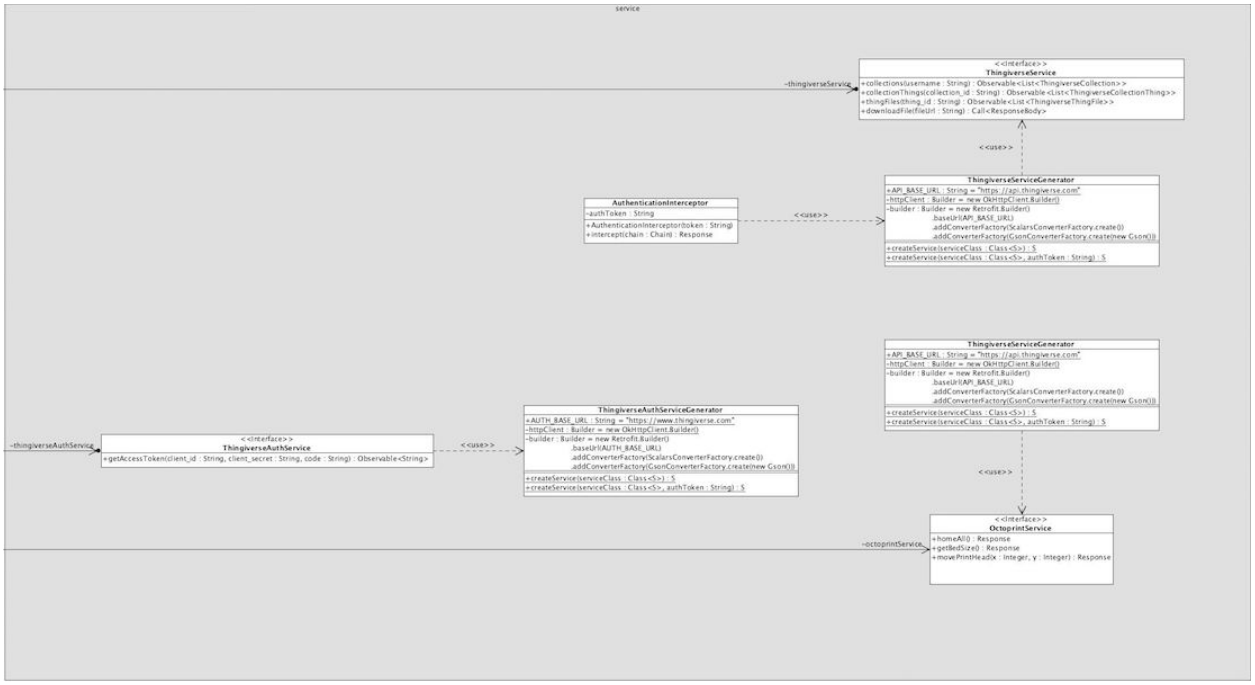
The View namespace is where all Activities within the application are defined. These Activities are very bare, and are merely vessels to convey data to the user - business logic is conducted within the ViewModel associated with the Activity.

5.2.1.4 ViewModel Namespace



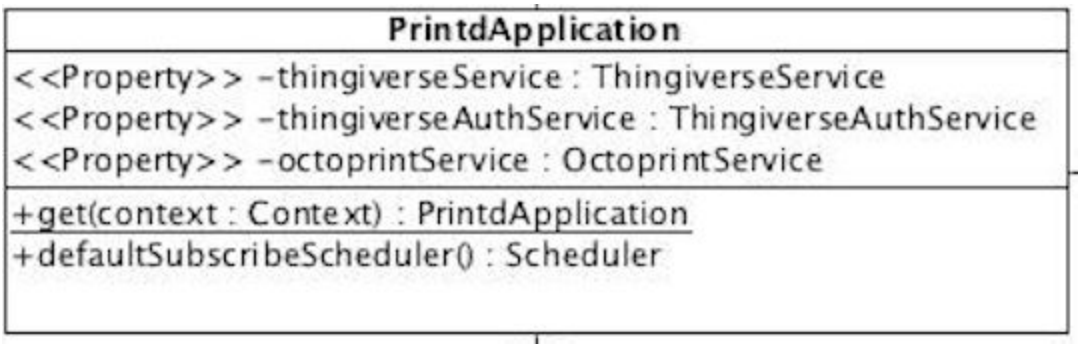
Each Activity has an associated ViewModel. These ViewModels contain the business logic of the application, and glue the Models to the View so the user can see data. Each ViewModel inherits from the base ViewModel interface, as seen in the diagram.

5.2.1.5 Service Namespace



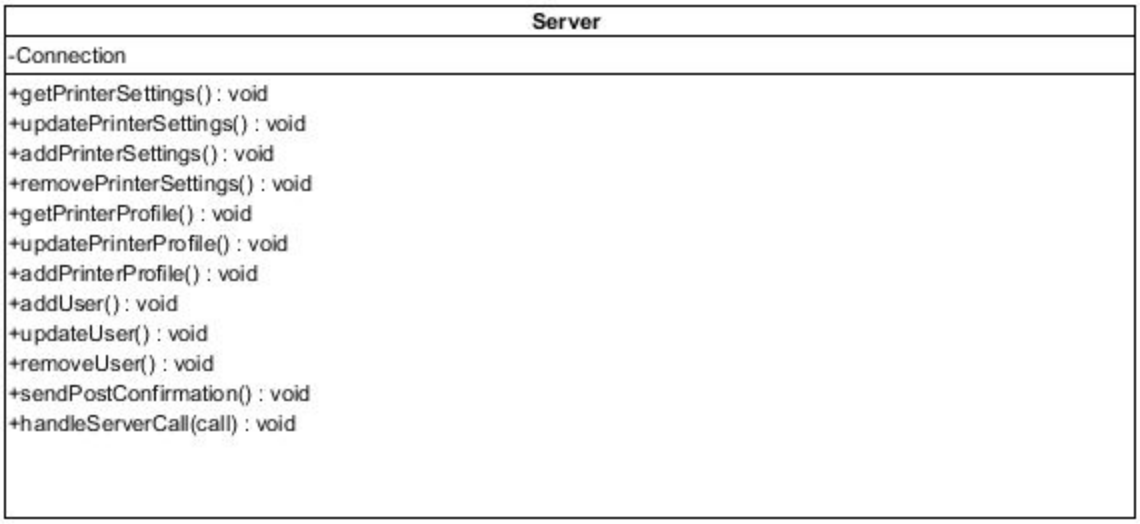
The Service namespace defines the external communication routes with APIs. Here, Generators and Interfaces are defined that the Retrofit library utilizes to create seamless interfaces that can be used throughout the application by obtaining them through the PrintdApplication Class.

5.2.1.6 Application Class



The Application Class is one of the most important classes in the application, and as such is not part of any of the namespaces. In it, other classes may access variables that are application wide, such as the Retrofit services that should only be defined once.

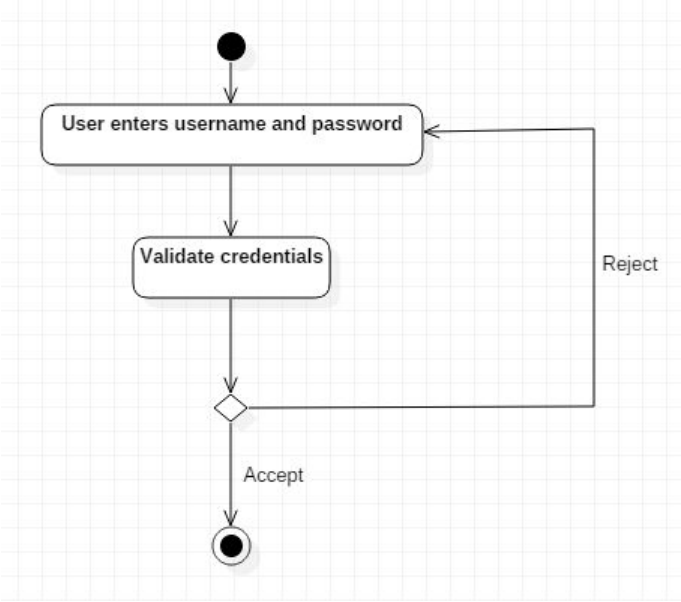
5.2.2 Server Class Diagram



The Heroku server has its own class diagram because it operates separately from the main application. The Heroku app is very basic, so will only have one class called Server. The connection with the database is stored in a Connection object. There is a handler method for server calls, which will then call the appropriate method depending on which server call is made. The actual database will be the Postgres database used with Heroku.

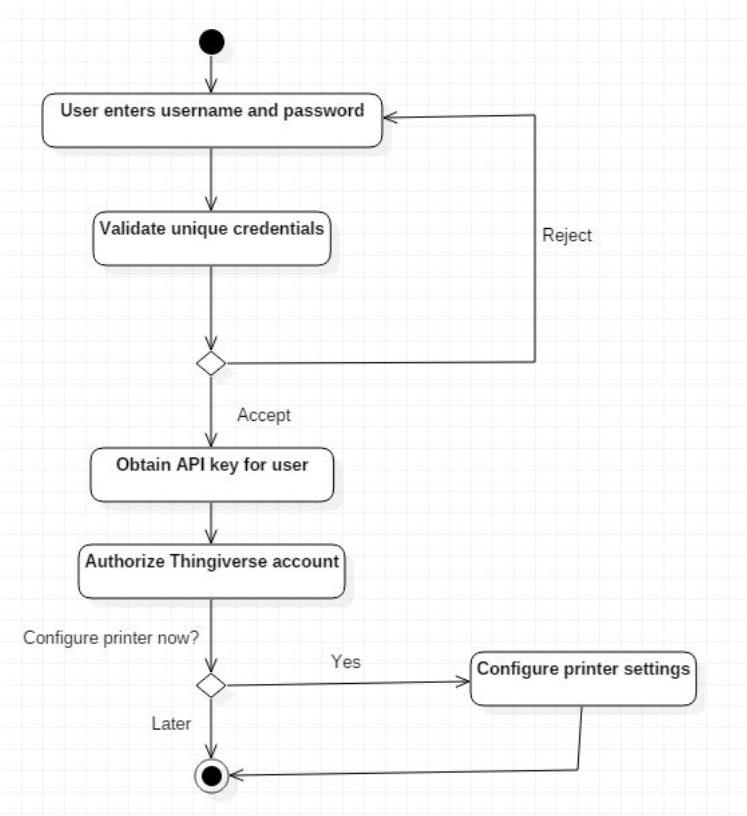
5.3 Activity Diagrams

5.3.1 Login



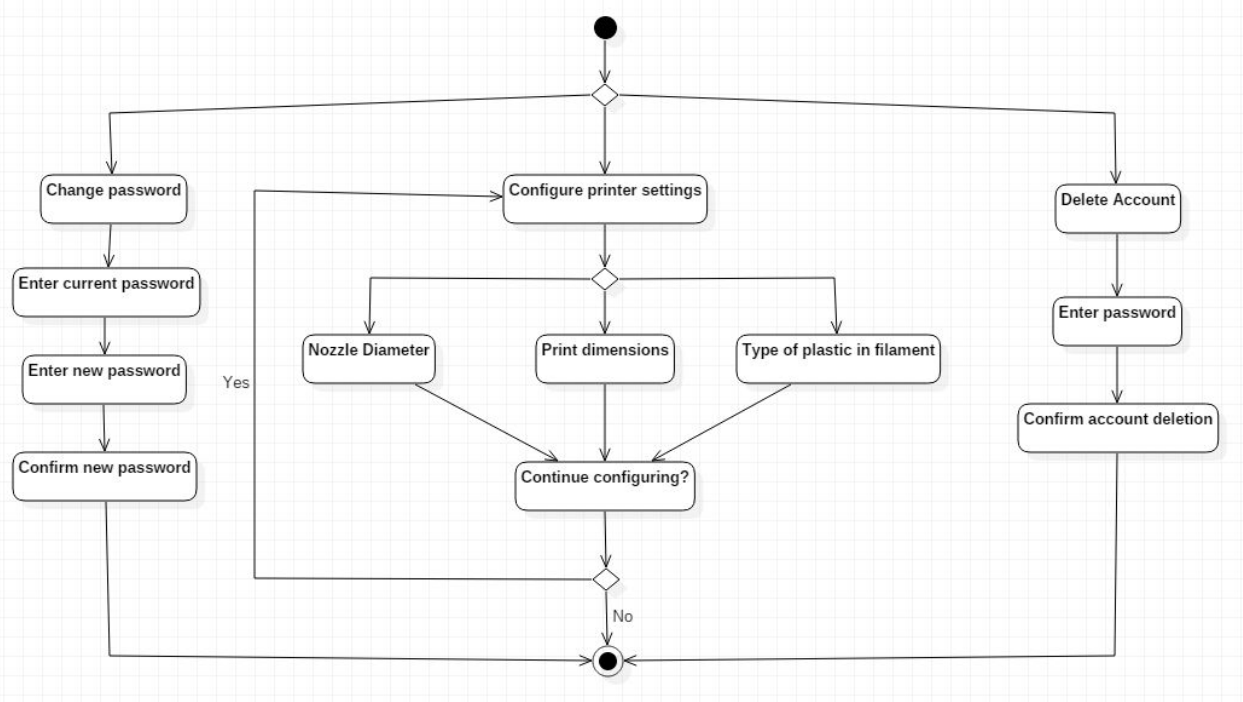
This is a diagram for a generic login activity. The user needs to enter their username and password and hit the login button. We will also have the option to remember their login info and log them in automatically when they launch the app. This could be a security issue so it is only recommended if the user has a passcode on their phone.

5.3.2 Create Account



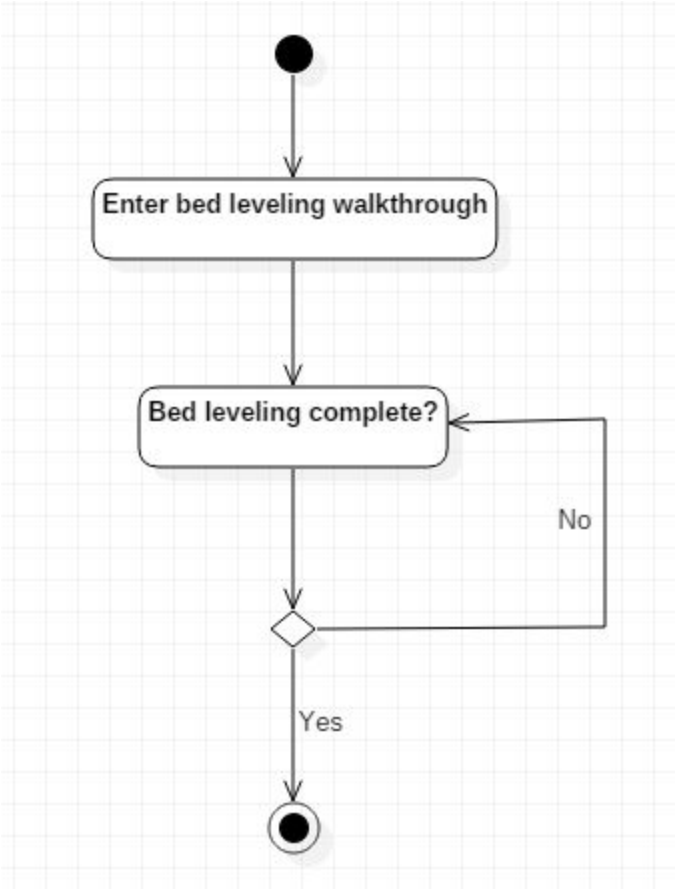
This is the activity diagram for creating an account. After the user creates a unique username and sets a password, they will be prompted to link their Thingiverse account. They are then presented with the option to configure their printer settings now or later. While it isn't necessary to configure the printer settings right away it is recommended.

5.3.3 Edit Account



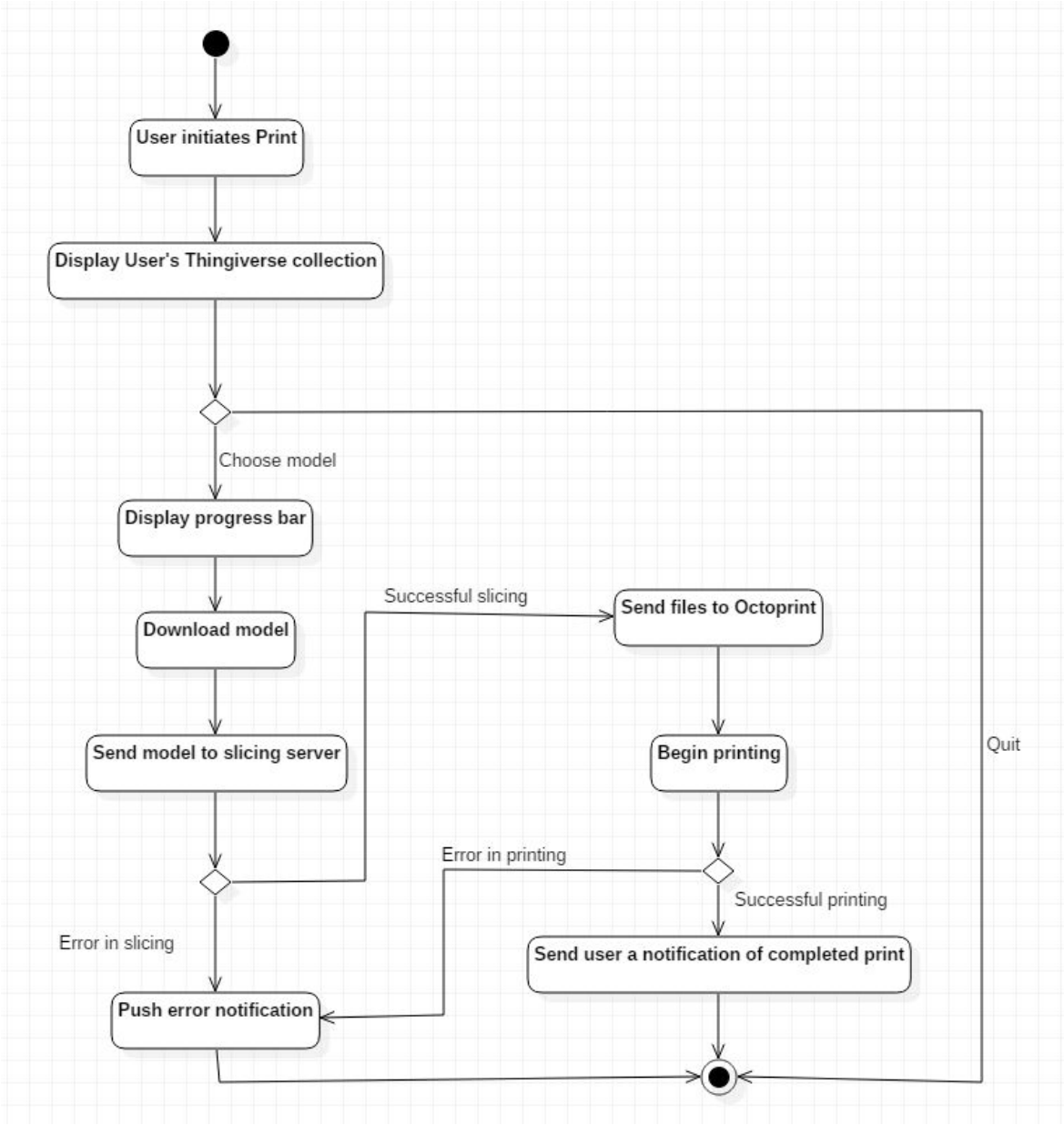
The activity for editing their account is pretty standard. They can do basic account management stuff like change their password or delete their account. The important part of this activity is that they can configure their printer settings here. This lets them put their printer model in, size of the bed, and type of filament. Ideally we will have a database full of printer settings created by users. This way new users can just select which type of printer they have and have all of the settings loaded in.

5.3.4 Bed Levelling



Having app-assisted bed leveling is pretty straightforward on the technical side. It is mostly just an instructional walkthrough on how to level the bed. The process is very tedious and this would help alleviate that.

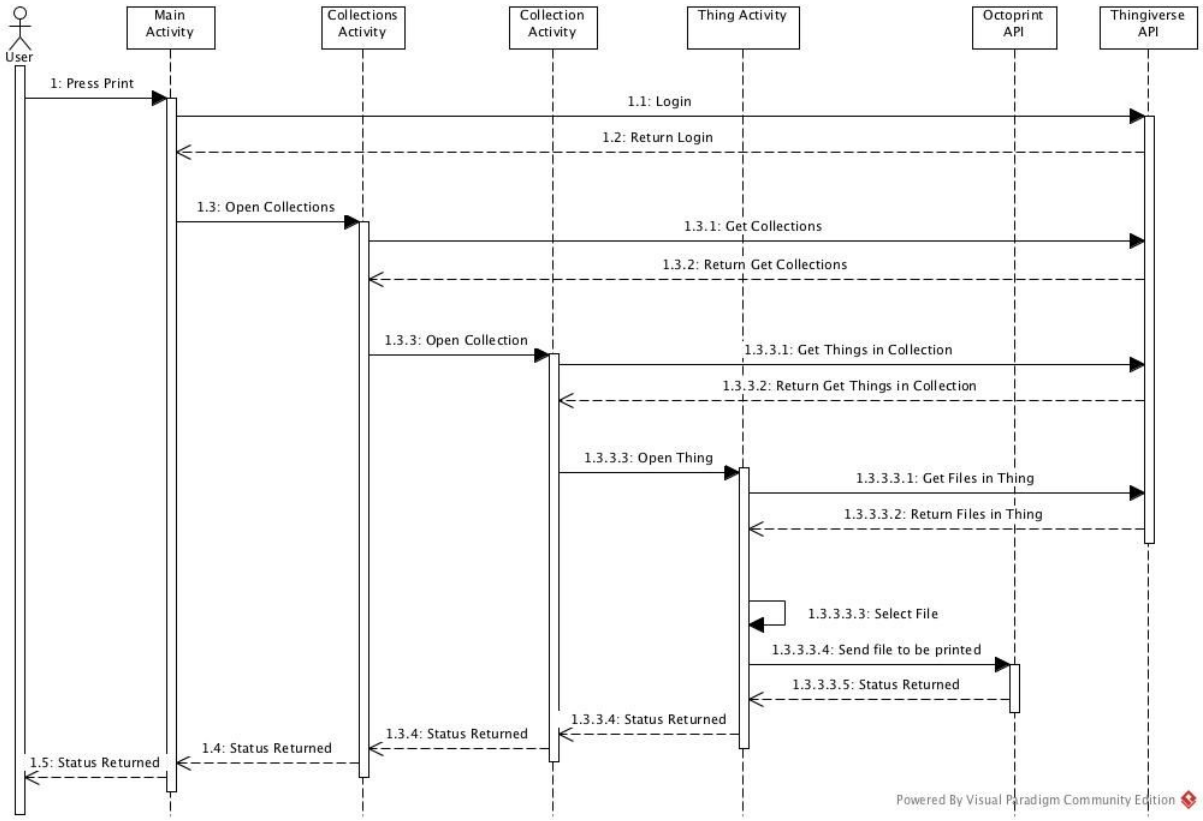
5.3.5 Printing



This is the main activity diagram as it covers printing. This highlights how much this app simplifies the process, since the user only has to open the app and choose a model to print. Since there are a lot of steps it is important to have a progress bar. Once the printer starts printing the progress bar can be changed to text saying that the print has begun. Since the printer may be left unattended in another room, the app will send a push notification to the user once printing has completed. Should the print fail at any point in the process, an error will be displayed to the user telling them at what step the app failed. There are many things that can go wrong when printing so it is important to be specific so the user can potentially fix the issue.

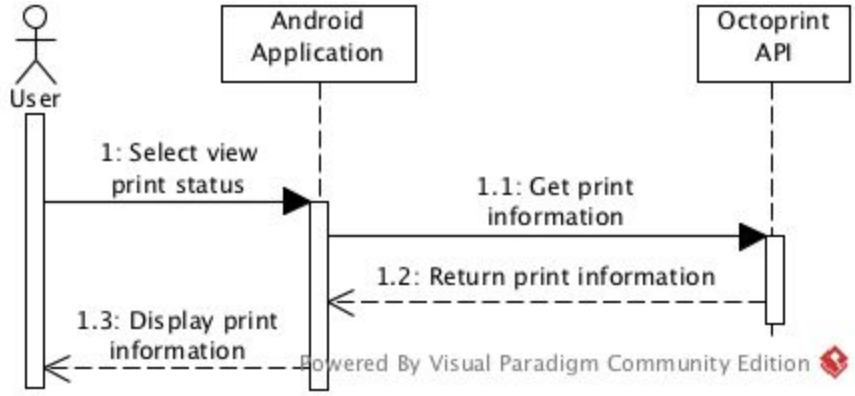
5.4 Sequence Diagrams

5.4.1 Print



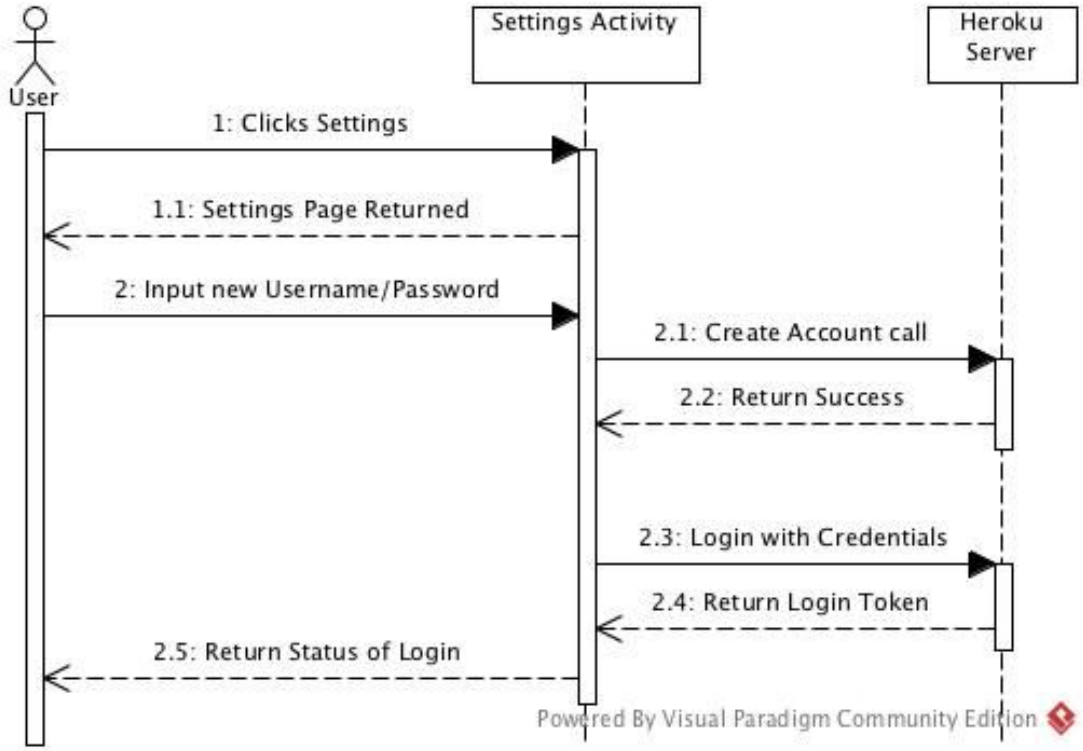
The Print sequence is the main flow of our application. In Print, the user telescopes through Thingiverse Activities that help them select a particular file to print. Each step of the way, calls are made to Thingiverse through their API to provide the user with the next step. Once a file is chosen, it is sent to OctoPrint to be sliced and printed, then the status of the print is returned to the user.

5.4.2 View Print Status



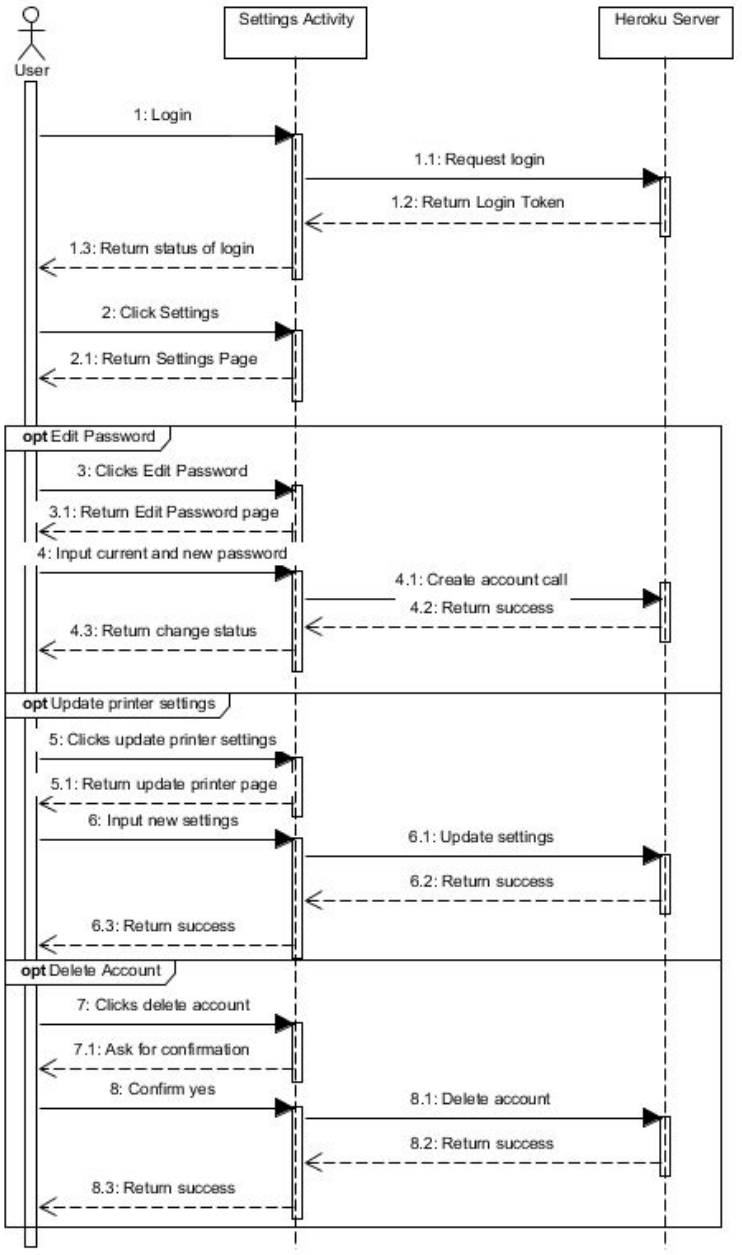
Users will be able to view the status of their print if their printer is currently printing a job. On the main page, the user will select the ‘View Print Status’ button, which will then send a request to the Octoprint API and display the relevant information, such as the percent completed, etc.

5.4.3 Create Account



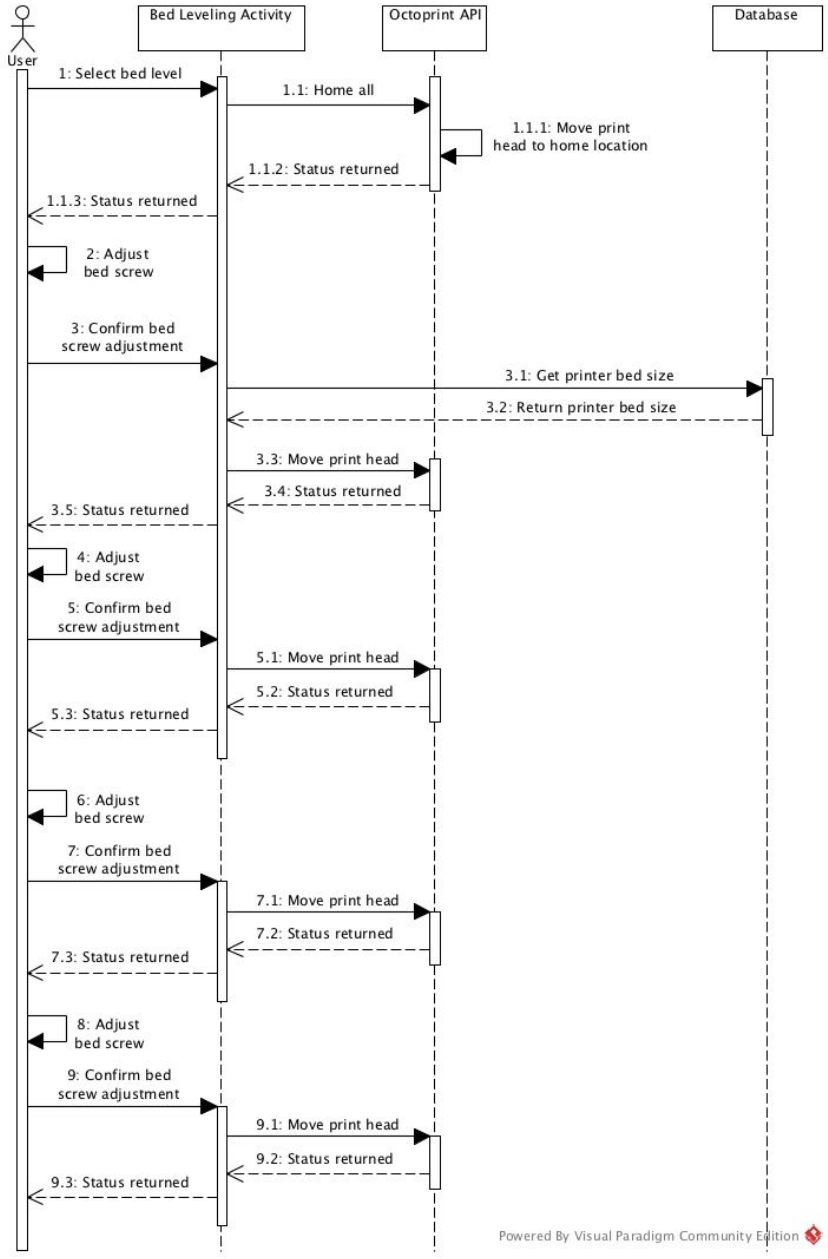
This diagram describes the steps taken by a user to create a new account. After settings is chosen, the settings activity returns the Settings page for the user to input new credentials. The settings activity then sends the credentials to the Heroku server. The server checks to make sure the credentials are valid, then returns a success. The activity continues to login in with the new credentials.

5.4.4 Edit Account



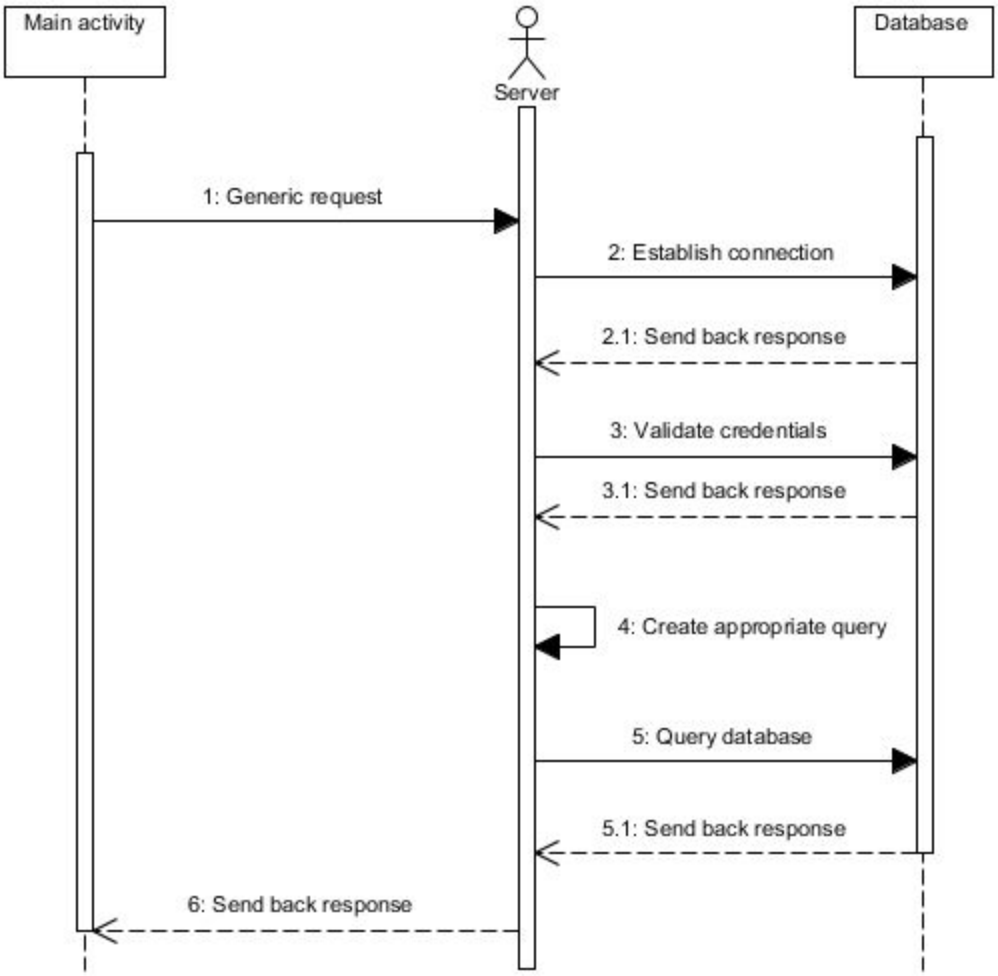
This diagram starts off with a login activity. Then the user selects settings to get to the edit account page. The three options currently are Edit Password, Update Printer Settings, and Delete Account. These options will present current settings to allow for updates. Then these changes are pushed to the Heroku server to update the data there.

5.4.5 Bed Leveling



The bed leveling activity is optional. If the user selects the bed leveling option, the application will send a request to the OctoPrint API to move the print head to the home location. Then, the user will be prompted to adjust the current bed screw. Once confirmed, the app will then get the printer bed size from the database to calculate the next print head movement. The print head will move, and the user will be prompted to adjust the next bed screw. This will be repeated for all bed screws, and then once

5.4.6 Server Call

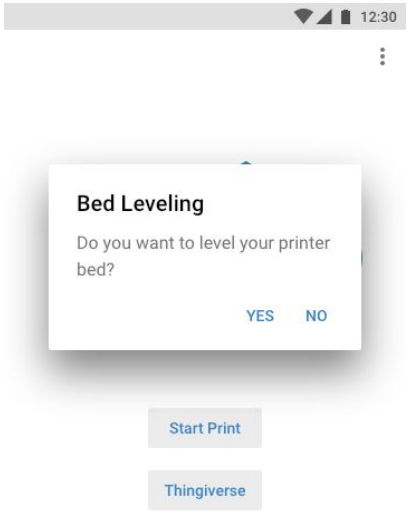


Since all of the server calls will be handled the same way, this is a sequence diagram for a generic server call. The main application sends a request to the Heroku server. The Heroku server then ensures it has a connection with the Postgres database, validates the user's credentials, creates the appropriate query, queries the database, and sends back the response.

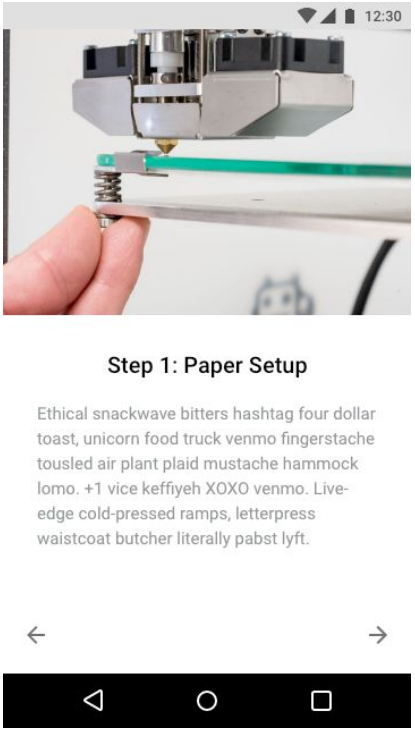
6. APPLICATION SCREEN MOCKUPS

Printing Status

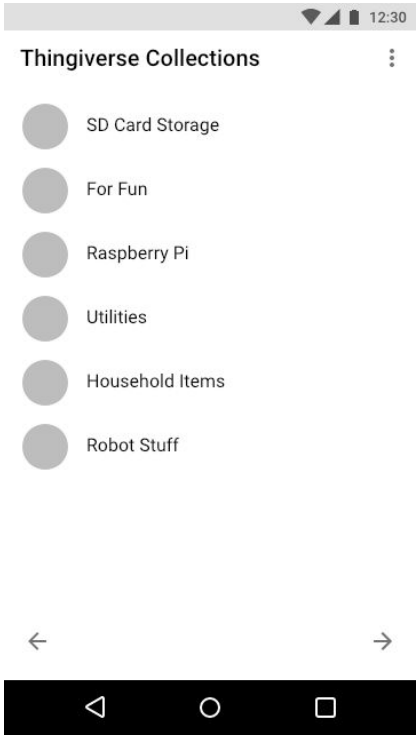
Bed Leveling Option



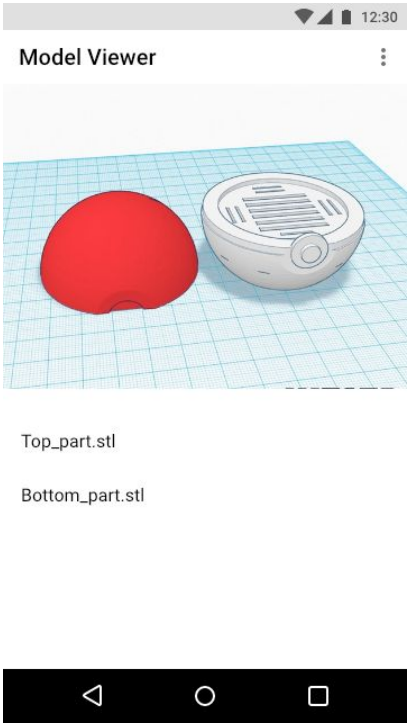
Bed Leveling Instructions



Thingiverse Collection Viewer



Model Viewer



Printing Status