Bilkent University

Department of Computer Engineering

Senior Design Project

Nino: Nino Is Not OCR

Analysis Report

Ata Deniz Aydın, Ecem İlgün, Ergün Batuhan Kaynak, Selim Fırat Yılmaz

Supervisor: Hamdi Dibeklioğlu

Jury Members: A. Ercüment Çiçek and Özcan Öztürk

Analysis Report
Nov 12, 2018

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1 Introduction **2**

2 Proposed system **2**

2.1 Overview . . . 2
2.2 Functional Requirements . . . 3
2.2.1 User Profile . . . 3
2.2.2 Taking Notes . . . 3
2.2.3 Note Visual Processing . . . 4
2.2.4 Note Text Processing . . . 4
2.2.5 Note Categorization . . . 4
2.2.6 Note Processing Feedback . . . 4
2.2.7 Note Exporting and Sharing . . . 5
2.3 Non-Functional Requirements . . . 5
2.3.1 Accessibility . . . 5
2.3.2 Accuracy . . . 5
2.3.3 Availability . . . 5
2.3.4 Backup and Recovery . . . 5
2.3.5 Capacity . . . 6
2.3.6 Compatibility . . . 6
2.3.7 Concurrency . . . 6
2.3.8 Configurability . . . 6
2.3.9 Error-Handling . . . 6
2.3.10 Extensibility . . . 6
2.3.11 Legal and Regulatory Requirements . . . 7
2.3.12 Licensing . . . 7
2.3.13 Maintainability . . . 7
2.3.14 Performance . . . 7
2.3.15 Reliability . . . 7
2.3.16 Scalability . . . 7
2.3.17 Security . . . 8
2.3.18 Testing . . . 8
2.3.19 Usability . . . 8
2.3.20 Portability . . . 8
2.4 Pseudo requirements . . . 8
2.5 System models . . . 10
2.5.1 Scenarios . . . 10
2.5.2 Use case model . . . 16
2.5.3 Object and class model . . . 17
2.5.4 Dynamic models . . . 20
2.5.5 User interface mockups . . . 28

3 Glossary **36**

1 Introduction

Note taking amounts to an important part of studying materials for many students, be they grade school or graduate students. In many cases, there are no lecture notes or slides readily available, or the ones that are available do not cover all of the lecture content. This requires students to take notes during the lecture.

However, in some cases, the student has to note down the words fast without truly processing their content, or write down equations without having the time to analyze and thus understand them. In the case of writing down sentences, one can achieve faster note taking with tablets/computers [1] but when the lecture notes consist of plots, graphs or mathematical equations, our market research shows that there is no hardware/software help to achieve faster note taking than writing it down [2].

In some cases, scribing all the things written on the board (even without making sense of it) proves to be quite hard, therefore many note taking strategies have arisen [3]. Many of these strategies revolve around the rule that one first takes notes of keywords, and then rewrites the notes after class [3]. However, there are two possible problems with this strategy: the student might forget the information conveyed via that particular keyword, or might miss important information and not write a keyword about it.

Currently, EverNote™, OneNote™ and Google Docs™ offer OCR (optical character recognition) features in which an image is converted to a text. However, these conversions do not cover mathematical equations or figures. As it is discussed above, many courses include some kind of figures or mathematics, therefore capturing only text with OCR might lead to loss of important information. Hence, current software are not capable of providing a complete note taking system.

We thus propose an app that will convert handwritten and printed notes into an editable format which can be exported as LaTeX or PDF. These notes will preserve the alignment of the original picture, in either landscape or portrait format. The student will be able to add his/her own notes on top of the ones extracted from the image. In this way, we hope to engineer a note taking system that would make the lecture not only more interactive for the student, but will also let the student listen to more of the lecture.

2 Proposed system

2.1 Overview

NINO (NINO Is Not OCR) will be an Android application, targeted primarily for tablets, that will be able to detect text, equations and figures in pictures taken of notes on paper or a whiteboard. The user will be able to take pictures directly within the application or load preexisting pictures from the device. After the program analyzes the picture, it will be able to present annotations on the picture and store them so that the user can later view and search

through them.

The application will consist of a client Android and web interface for end users, as well as a backend server application for data storage and processing. In the client application, the user will first be presented with a login screen where the user may register or log in. After logging in the user will be able either to add a new picture to their library, whether by camera or from the device, or to view previously annotated pictures. After a picture is taken it will be sent to the server for processing. The server will first segment the image into regions containing text, equations, graphs or other figures, parse the text or equations (in LaTeX) contained in each region, and then process the text and equations in order to provide helpful hyperlinks to the user. For example, the program may link a variable or concept to where it was defined in the notes, or to an online resource such as the Wikipedia page for the concept which the user would be able to open without leaving the application. The program will also be able to process the text in the document to summarize its content and tag it with keywords in order to enable more comprehensive searches.

After processing the server will store the annotated picture in the cloud storage reserved for the user and send it back to the client application. The client will then display the notes overlaid on the image in an interactive interface where the user may click on links, rearrange or modify the notes as desired. The user may give positive or negative feedback on the performance of the recognition of text and equations, and opt to make corrections and send them to improve the accuracy of the program. The user will also be able to share the annotated picture either directly as an image or as a LaTeX [4] or PDF [5] file. The notes the user has uploaded will be collected within the personal storage space allotted to the user, so that the user can view and modify them similarly, search through them, and categorize them e.g. with respect to course title and subject so as to assist the program in improving its tagging and summarization of notes.

2.2 Functional Requirements

2.2.1 User Profile

- Anyone will be able to sign up for the application by providing their name and phone number.
- During registration, verification SMS will be sent to given phone number.
- The user will be logged in automatically for subsequent openings of the application.
- The user will be able to access the original photos and processed versions of their notes.
- The user will be able to duplicate previously created notes.

2.2.2 Taking Notes

- The user will be able to take pictures of notes from the application.

- The user will be able to import pictures of notes from gallery to the application.
- After taking a note, the user may manually assign the note to a category, or the program may suggest a category for the note. If no category has been created, the note will be assigned to a default category.
- The user will be able to create new categories, rename previously created categories, and delete previously created categories (except the default category).
- The user will be able to change the category of a note.

2.2.3 Note Visual Processing

- Texts in the image will be recognized and converted to ASCII format.
- Equations in the image will be recognized and converted to TEX format.
- Shapes and plots in the image will be recognized and converted to TEX format, in the form of TikZ shapes and plots [6].
- Faces of people in the original photo will be blurred.
- Taken or imported photos will be processed automatically and their LATEX and PDF versions will be stored. Thus, for each note .jpeg (the original blurred photo), .tex, and .pdf files will be stored in memory.

2.2.4 Note Text Processing

- Texts of notes will be summarized, namely important keywords will be extracted.
- Terms in the texts (such as photosynthesis) will be recognized and matched to their Wikipedia link and description.

2.2.5 Note Categorization

- Each category and each note will be profiled, by associating to each a profile vector (a profile vector will be created for each of them).
- Each newly taken photo will be suggested a category according to its profile, by associating it with the category with the nearest profile vector (Nearest category profile vector will be found and its category will be suggested).

2.2.6 Note Processing Feedback

- User will be able to give feedback for mistaken analyses to the processing system and system will try not to do same mistake again (by means of reinforcement learning).

2.2.7 Note Exporting and Sharing

- User will be able to export notes as LaTeX or PDF files.
- User will be able to export and share single or multiple notes as .png, .tex, or .pdf files, via e-mail or WhatsApp if possible.
- User will be able to obtain a shareable link to a note.
- Users and non-users will be able to access notes via shareable links.

2.3 Non-Functional Requirements

2.3.1 Accessibility

- The system will need the input documents to be in English for all desired functions to work. User interface will have many language translations added on demand, starting with English as default.

2.3.2 Accuracy

- All functionality must work in a way that the results are at least acceptable. No output should look like random guessing. As such the accuracy of the program will be validated and tested on designated datasets, and ensured to stay above a predetermined threshold.

2.3.3 Availability

- System uptime should be high to not cause any inconvenience to the user.
- Scheduled maintenances to the core functionality should be announced beforehand and should not take very long.
- Since the system is highly modular, the whole system should not be down due to an update to a particular submodule.

2.3.4 Backup and Recovery

- The users should have the ability to locally backup their work.
- The server should do regular backups of user data and preferences in a certain time window.
- Data protection measures such as RAID may be used in order to recover at least partially from data corruption, without compensating too much from performance.

2.3.5 Capacity

- The server must have sufficient sufficient storage / compute for each user.
- The server must have sufficient sufficient storage / compute to handle at least 10.000 registered users and up to 100 concurrent requests at launch.

2.3.6 Compatibility

- Client user interfaces for web and Android should be compatible with a wide range of browsers and Android versions, enough to statistically support a good percentage of potential users.
- Popular output formats, such as LaTeX and PDF, should be an export option.

2.3.7 Concurrency

- The server should be able to handle requests in an either a concurrent or parallel manner, the users should not have to wait for other requests to be handled in a ?first in first out? type of sequential manner.
- Up to 100 request at any given time should be accepted for processing by the server.

2.3.8 Configurability

- The users should be able to manually modify the format of the output, module-specific properties, or the overall end product itself.

2.3.9 Error-Handling

- Errors must be recognized and accounted for as much as possible. The user should see an error message that clearly and concisely explains what went wrong to understand what the problem is without being presented with too much needless info, with steps to take to solve the problem, if the nature of the problem permits.
- Any unforeseen errors should be presented to the user with an appropriate message to the user, with an option to the user to report the problem to the developers.

2.3.10 Extensibility

- The system will be highly modular. Any additional modules will be integrated without causing any problems to other modules. The system will be implemented in a way that addition of extra modules will be easy, so long the module is written.

- Adding extra modules to the system should be straightforward as long as the module functions correctly by itself.

2.3.11 Legal and Regulatory Requirements

- The users should be made aware that the developers do not take any responsibility on the event that the user uses the app to violate any laws (e.g. copying licensed material, editing material etc.) and that the legal responsibilities lie solely on the user, not the developers.
- Legal regulations should be presented to the user with appropriate disclaimers and necessary citations.

2.3.12 Licensing

- Licenses of any products, libraries, or modules used during development should be adhered to.

2.3.13 Maintainability

- Since the system is highly modular, subsystems should not be highly coupled, a change or problem in one subsystems should not affect the others.

2.3.14 Performance

- Each function module must not take more than 5 seconds to complete its process.
- Results of any human interaction (other modules) such as clicks or selections should not take more than 1 second to provide good user experience.

2.3.15 Reliability

- If any of the modules use means that cannot be configured by the developers to be more reliable and that can cause potential problems, then the potential problems that can occur while using such modules should be presented to the user.
- The system must use learning techniques to adapt to different situations both in general and for each user to give the best possible results.

2.3.16 Scalability

- Adding more storage and compute to the server should be easy as long as new storage / compute hardware is available.

- Rise in demand must be addressed quickly to not cause any drop in performance or downtimes in the system.
- Modules should be able to work on more than one document if available to make better predictions.

2.3.17 Security

- Users must authenticate themselves before using the system.
- No user should be able to access documents or preferences of other users.
- User login data should be hashed to make sure they are safe even in an event of security breach in the server.
- Any number of requests that points to bot action or DDOS should be prevented by taking appropriate measures.

2.3.18 Testing

- Each module / component should be subject to at least unit testing to make sure there is a good coverage of potential problems before deployment.

2.3.19 Usability

- Users should be able to navigate through the clients without having problems on how to use it, the process should seem natural without a high learning curve.
- There should be appropriate description to each module so that the user can clearly understand its function and use cases, preferably with examples or tutorials.
- User interfaces for web and Android should not differ too much, so that they do not seem like altogether different products for users switching devices.
- Users should be able to contact developers to make suggestions or report problems and their experience.

2.3.20 Portability

- The user client should start with a web client. The user interface should be able to be ported to Android.

2.4 Pseudo requirements

- Git will be used for version control.

- The application will be developed under the paradigm of object-oriented programming.
- External libraries will be used to train and run neural networks for image and natural language processing.
- In order to blur faces in images, the program will utilize an external API for face recognition.
- The system will be implemented in a modular fashion for easier testing and maintenance, with different modules to segment document images into text, equations and figures, to detect text, to parse equations, and to process text for annotation and summarization.
- The system may later be extended with more modules without affecting the performance of preexisting modules.
- For training and testing we will firstly use our own notes, as well as handwritten course notes and blackboard pictures available online. Stills from lecture videos may also be used with appropriate permission or credit, depending on license. For symbol recognition datasets from previous research projects [8] are available.
- Costs may be incurred to maintain servers and to publish the app in the Google Play Store.
- These costs we intend to handle by ourselves for now, without selling the app for a cost or including advertising.
- User data will be encrypted, accessible only after authenticating, and will not be shared with anyone else or used for training without the user?s explicit consent.
- Licenses for the datasets and APIs used will be checked and appropriate permission will be sought before they are used in the project.
- Users may provide feedback on the accuracy of the text or equations recognized, and offer their corrections to the annotations produced.
- Tested third-party APIs with certain performance guarantees will be used for text detection.
- Modules will be tested first by themselves and then in tandem during implementation or later retraining.
- Consent of the content owner is required. However, since we cannot ensure whether consent is taken before taking picture of a material, we do not take any responsibility beyond this. Taking consent for any kind of material to be uploaded to our servers are the user's responsibility.
- Since we will store the content in our servers, we will provide security for the user?s data, and we will not share this data with third parties. However, the user will be able to share the content and processed version so it is expected from user to abide by copyright of the content.

- People's faces in the stored picture will be blurred to avoid any legal issues.
- Permission will be sought if required from content owners of datasets used for training and testing, such as lecture videos. In case videos are marked with a Creative Commons license [7] giving credit to the content owners will suffice.
- Licenses for third-party APIs and libraries will likewise be checked before usage.

2.5 System models

2.5.1 Scenarios

Scenario 1 Sign Up

Actors: New User

Entry Conditions: User opens the app

Exit Conditions: User is navigated to their personal homepage.

Flow of Events:

1. User clicks "Sign Up" button on the Login page.
2. The registration page is opened.
3. User enters their name and phone number.
4. User clicks "Sign Up" button.
5. An authentication SMS is sent to user's phone.
6. User enters the numbers given in the SMS.
7. User clicks "Verify the Code" button.
8. System verifies the SMS code.
9. System creates an account for the user.

Scenario 2 Login

Actors: Registered User

Entry Conditions: User opens the app

Exit Conditions: User clicks "Back" button of their phone and is redirected to their phone's homepage.

Flow of Events:

1. If User logged in from the device before the flow of events:
    - System matches User's cached credentials from the database.

2. Else the User enters their credentials and clicks "Login" button.
    - System matches the credentials from the database.
3. System logs the user in, and their account homepage with their previous notes are opened.

Scenario 3 Create Note

Actors: User

Entry Conditions: User is in their account homepage

Exit Conditions: User views the folder they are navigated to.

Flow of Events:

1. System provides the user with the notes they had previously taken.
2. If user clicks on a folder, system shows contents of the folder.
3. User clicks "Create Note" button.
4. "Add Photographs & View PDF Version" Scenario is invoked.
5. If user clicks "Done":
    - A note is created in the folder User was in before they clicked "Create Note" button.
    - System prompts the closest category to the User.
        - User chooses that category for the note.
            * Note is moved to the folder of that category.
6. User is navigated back into the folder they were last in.

Scenario 4 Add Photographs & View PDF Version

Actors: User

Entry Conditions: User clicked "Add Page" when viewing a Note or User clicked "Create Note" inside a folder.

Exit Conditions: User finished adding pages and views the PDF version of the document.

Flow of Events:

1. User is prompted with "Take Photos" and "Import from Gallery" options.
    - If User chooses "Import from Gallery", they choose photographs from their phone's gallery.
    - If User chooses "Take Photos", they take any number of photographs of handwritten or digital notes.
2. User is done taking the notes and clicks "Next" button.

- System detects and blurs any face in the photographs.
- System detects the margins of the notes in the photographs.

3. System shows the user the detected margins.
4. User edits the margins.
5. User clicks "Next" button.
    - System detects & recognizes Mathematical Expressions, Texts and Sketches (such as plots, charts, cartesian system, graphs, trees, diagrams etc.).
    - System converts the recognized formats into their digital version (e.g. a plot will be generated digitally).
    - System keeps the unrecognized formats as images.
    - System arranges the segments into a PDF document, each photograph makes a page.
6. User views the PDF version of the document.
7. If user clicks "Discard" button during any of the Steps:
    - All previous changes are discarded.

Scenario 5 View Note

Actors: User

Entry Conditions: User is in their account homepage

Exit Conditions: User clicks "Back" button of their phone and is redirected to their account's Homepage

Flow of Events:

1. System provides the user with the notes they had previously taken.
2. If user clicks on a folder, system shows contents of the folder.
3. User clicks on one of their notes.
4. System provides the user LaTeX generated pdf version of the note.
5. User clicks on "Options" button.
6. User clicks on "View Original Images"
7. System provides the user face-blurred JPG image version of the notes.
8. User clicks on "Options" button.
9. User clicks on "View PDF version".
10. System provides the user LaTeX generated pdf version of the note.
11. User clicks on a keyword in the PDF version.

- System searches Wikipedia for a page of the keyword.
- If the keyword's page exists, a pop up to that page appears.
  - User clicks "Back" button of their phone.

Scenario 6 Edit Existing Note

Actors: User

Entry Conditions: User clicks on one of the notes

Exit Conditions: User clicks "Back" button of their phone and views the PDF version of the note

Flow of Events:

1. System provides the user LaTeX generated pdf version of the note.
2. User clicks on "Edit" button.
3. System provides the user with an editable format of the note.
4. User clicks once on a segment.
   - User translates/moves each segment of Texts, Mathematical Expressions, Sketches in the document as they wish.
5. User clicks twice on a segment.
   - A popup screen of the segments digitized version appears. (e.g. a digital text)
   - User clicks "Delete" button.
   - System deletes the segment from the PDF version of the note.
   - Pop up screen disappears.
6. User clicks twice on a Text or Mathematical Expression segment.
   - System provides a pop up screen of the digitized version of the given segment.
   - User edits the text/mathematical expression the same way they would in a word document.
   - User clicks "Done".
   - Pop up screen disappears.

Scenario 7 Add Image to Existing Note

Actors: User

Entry Conditions: User clicks on one of the notes

Exit Conditions: User clicks "Back" button of their phone and views the PDF version of the note

Flow of Events:

1. System provides the user LaTeX generated pdf version of the note.
2. User clicks on "Edit" button.
3. System provides the user with an editable format of the note.
4. User clicks "Add Page" button.
5. "Add Photographs & View PDF Version" Scenario is invoked.
6. If user clicks "Done":
    - The new pages are added to the note's both JPG and PDF versions.
7. System provides the user with an editable format of the note.

Scenario 8 Arrange into Folders/Categories

Actors: User

Entry Conditions: User is in their account homepage

Exit Conditions: User finished arranging folders and is in their account homepage.

Flow of Events:

1. System provides the user with the notes they had previously taken.
2. If user clicks on a folder, system shows contents of the folder.
3. User clicks "Create Category" button.
4. System provides user with a pop up to enter the name of the category.
5. User enters a name for the new category.
6. System creates a new folder in the given directory.
    - User taps on any number of the notes.
7. System selects the note for further action.
8. User selects "Cut" button.
9. User clicks a folder.
    - System navigates User into the new folder.
10. User clicks "Back" button of their phone.
    - System navigates User into the parent folder.
11. If user clicks "Paste":

- The cut notes are pasted into the given directory.

12. User views the directory they pasted the notes to.

Scenario 9 Duplicate/Export Note

Actors: User

Entry Conditions: User clicks on one of the notes

Exit Conditions: User clicks "Back" button of their phone and is navigated to the category the note is assigned to

Flow of Events:

1. System provides the user a LaTeX generated PDF version of the note.
2. User clicks on "Duplicate" button.
    - System prompts a name for the duplicate.
    - User enter the name.
    - System creates a duplicate in the same folder/category.
3. User clicks "Export" button.
    - A pop up page with a URL link and modes of export (e-mail, WhatsApp, ...) appears.
    - User clicks on "Copy the URL".
    - User clicks on the mail button to export via e-mail.
        - User is forwarded into the respective e-mail app with the note attached to a draft mail, in .tex, .jpg and .pdf forms. User can then choose to keep or delete some of the forms and send one of them.
        - Once the User sends the e-mail, they are redirected to viewing the PDF format of the note.

2.5.2 Use case model

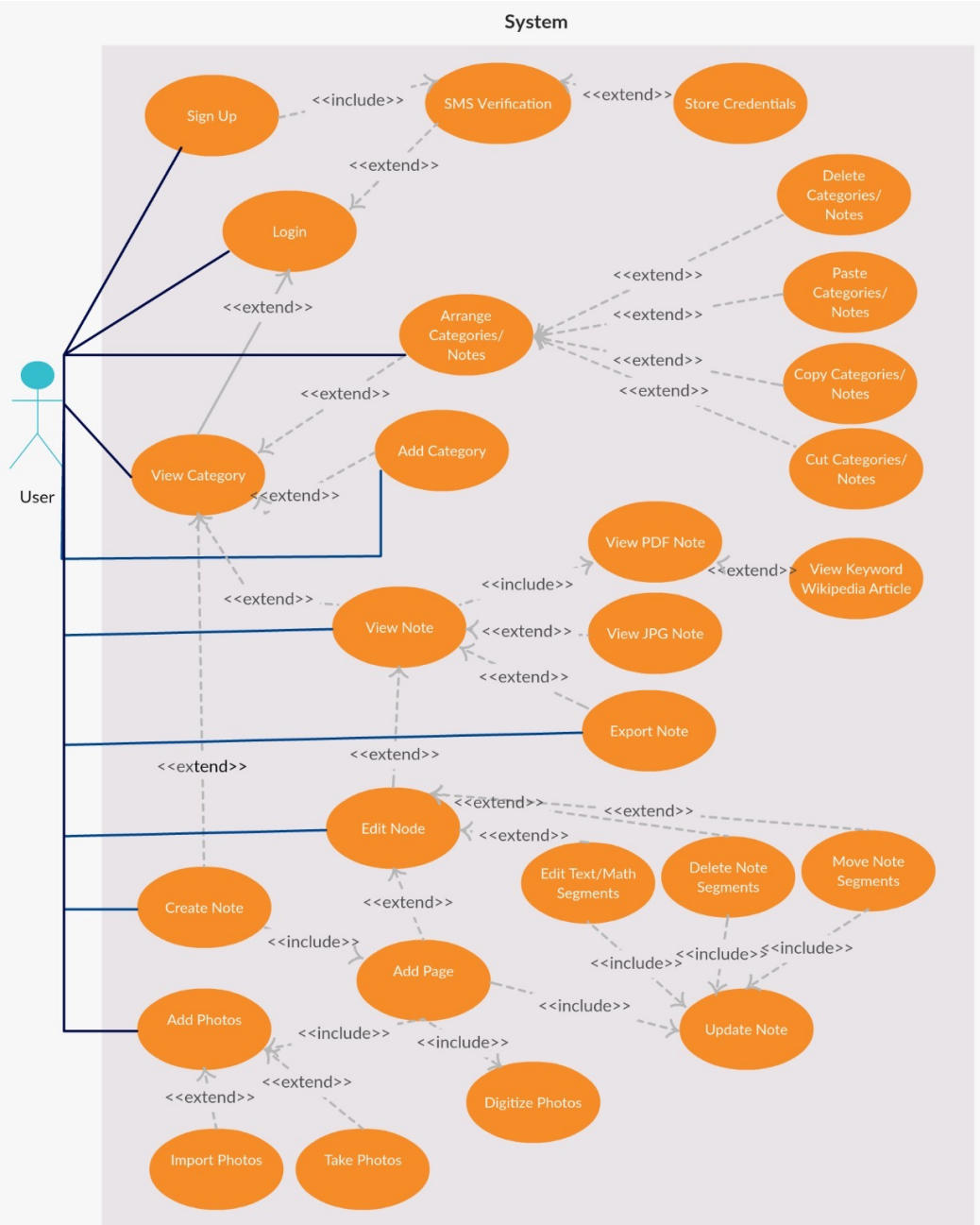

Figure 1: Use case diagram for the aforementioned scenarios.

2.5.3 Object and class model

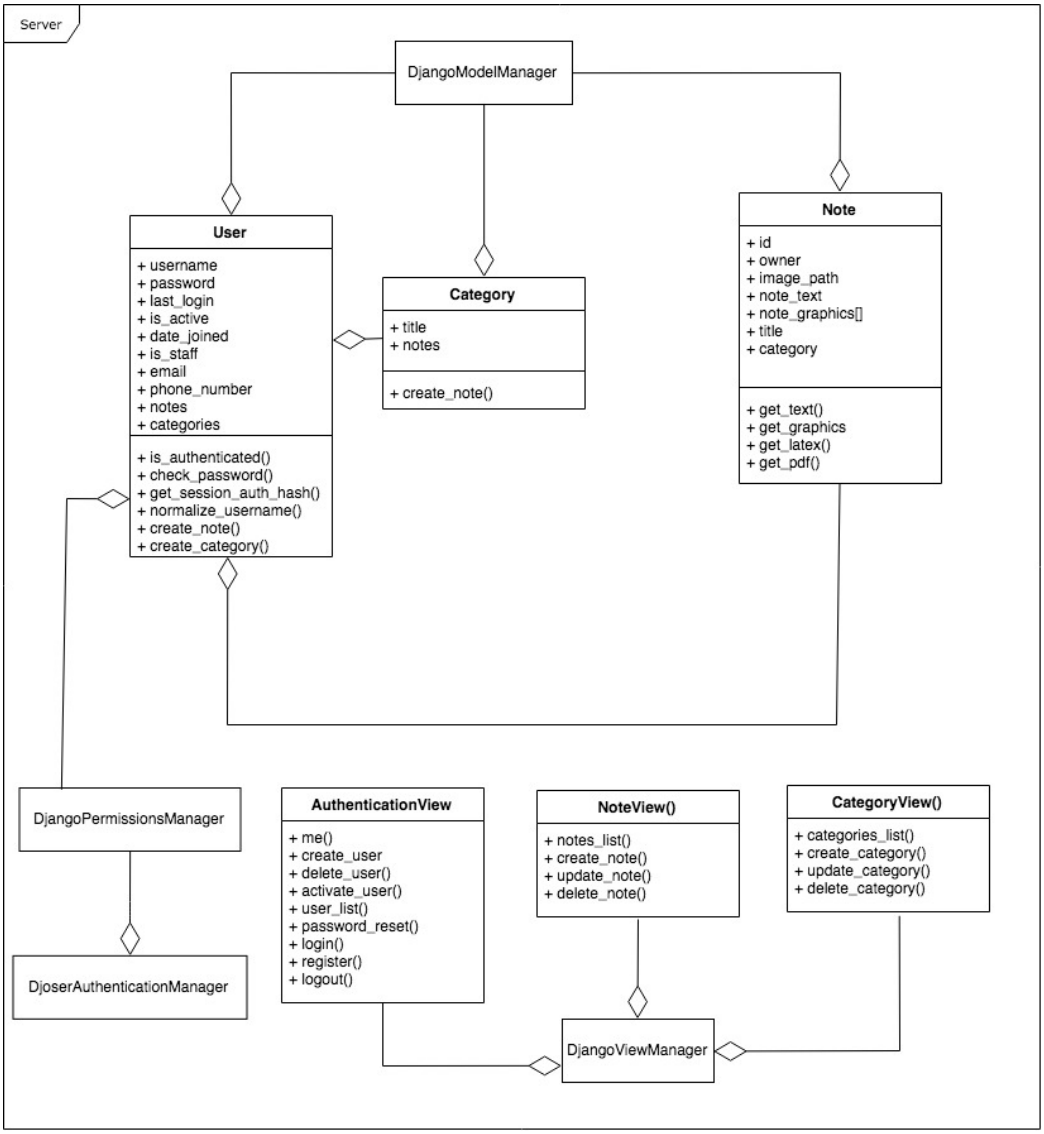

Figure 2: Class diagram for client-side objects.

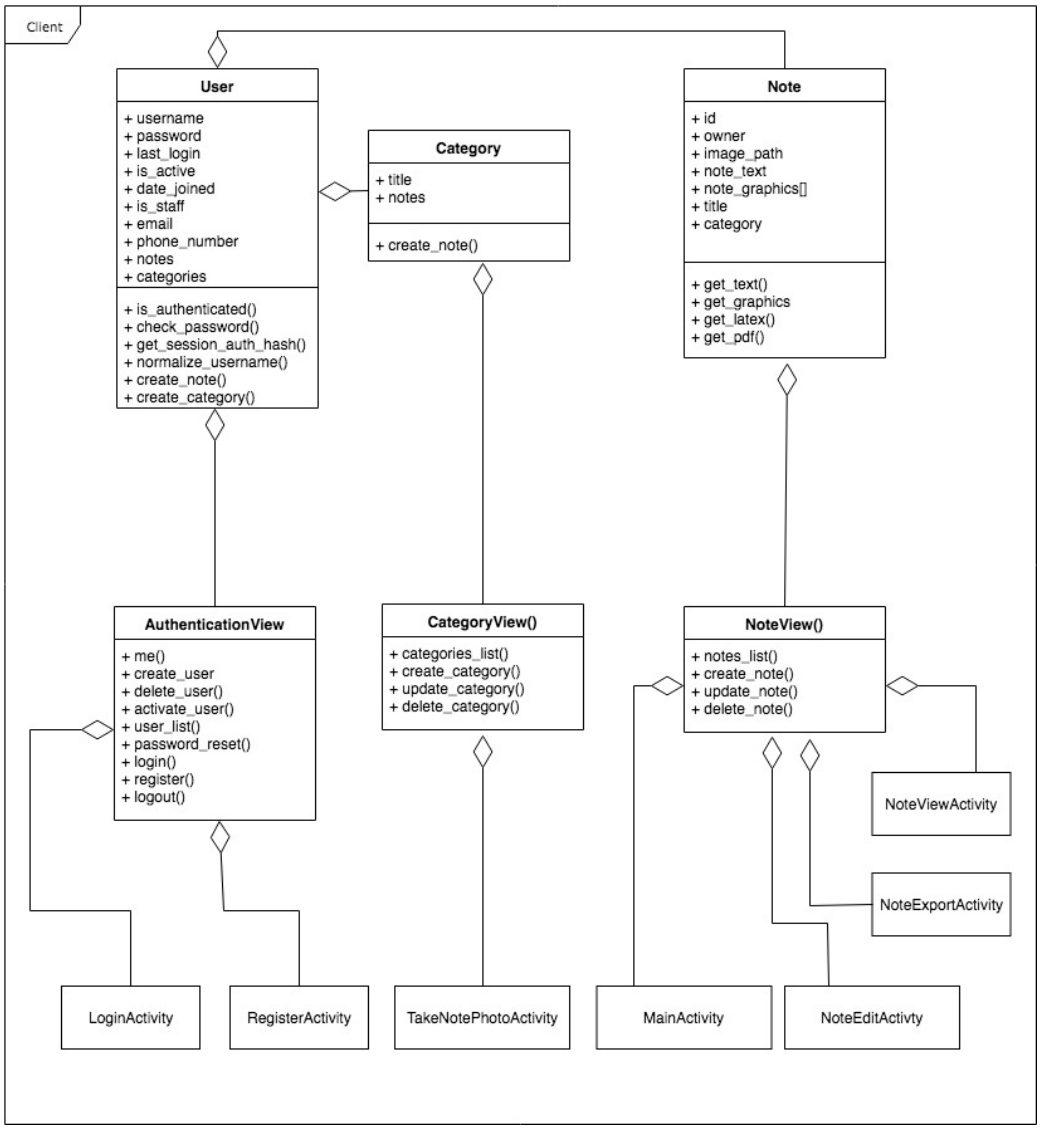

Figure 3: Class diagram for server-side objects.

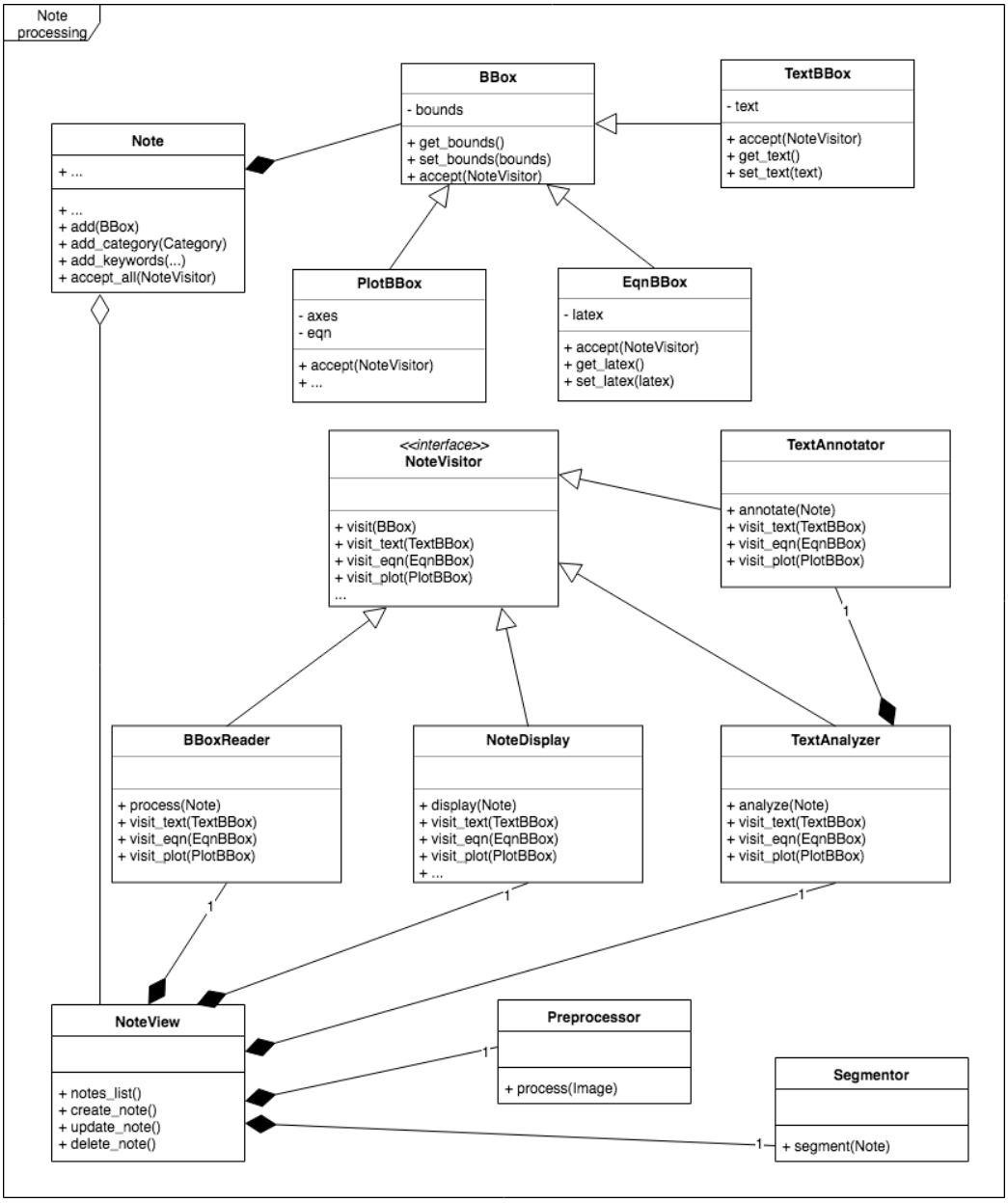

Figure 4: Class diagram for objects for generating, analyzing and displaying notes.

2.5.4 Dynamic models

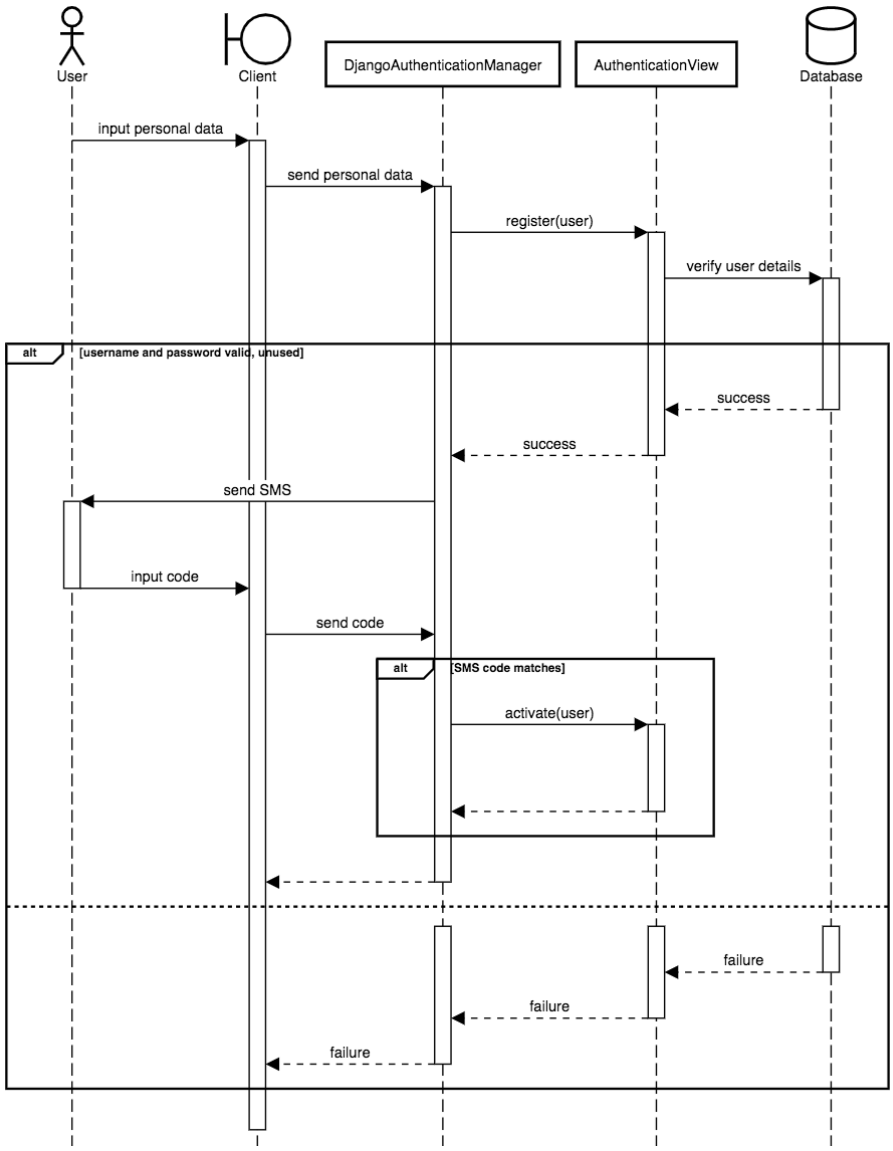

Figure 5: Sequence diagram modeling the registration of a new account. The user first inputs personal login data, and after they are validated he is prompted for SMS authentication, and afterwards the new account is activated if the authentication is successful.

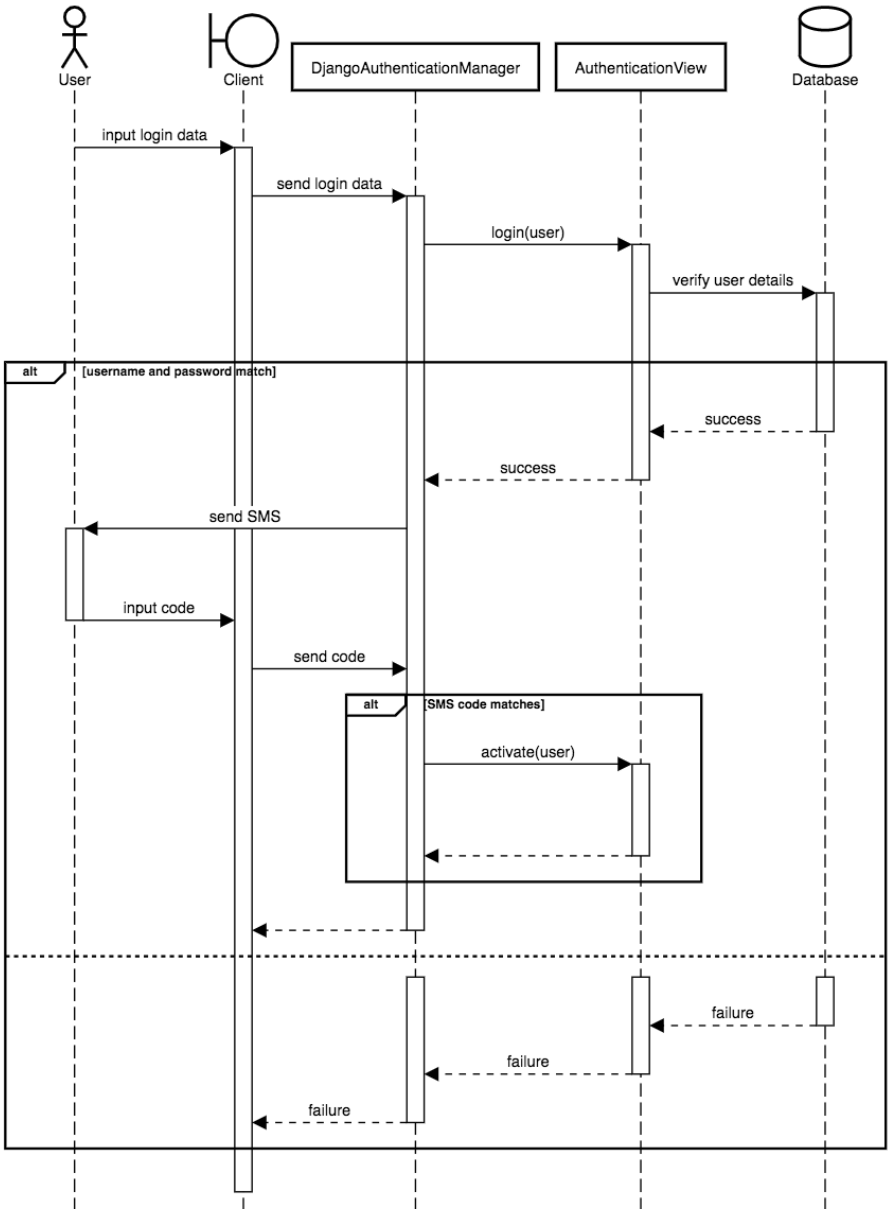

Figure 6: Sequence diagram for an existing user logging in. The user first enters login data, and if they are validated he is asked for SMS authentication, and logs in successfully if he enters the correct code.

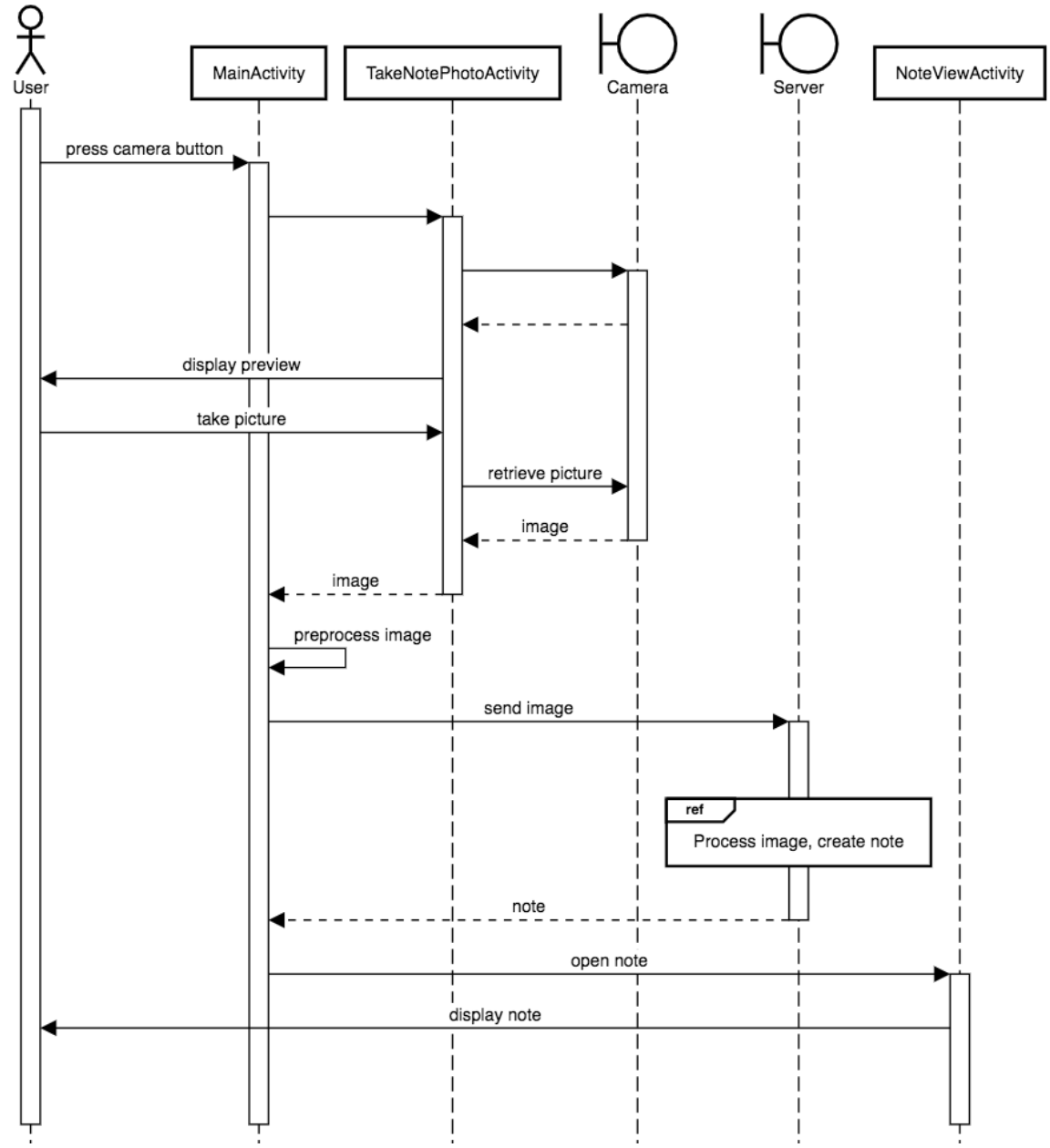

Figure 7: Creating a new note from camera: the user first takes a picture from the camera, then the device preprocesses it (e.g. removes faces from the image), then sends it to the server to be processed, and the server sends back the note produced.

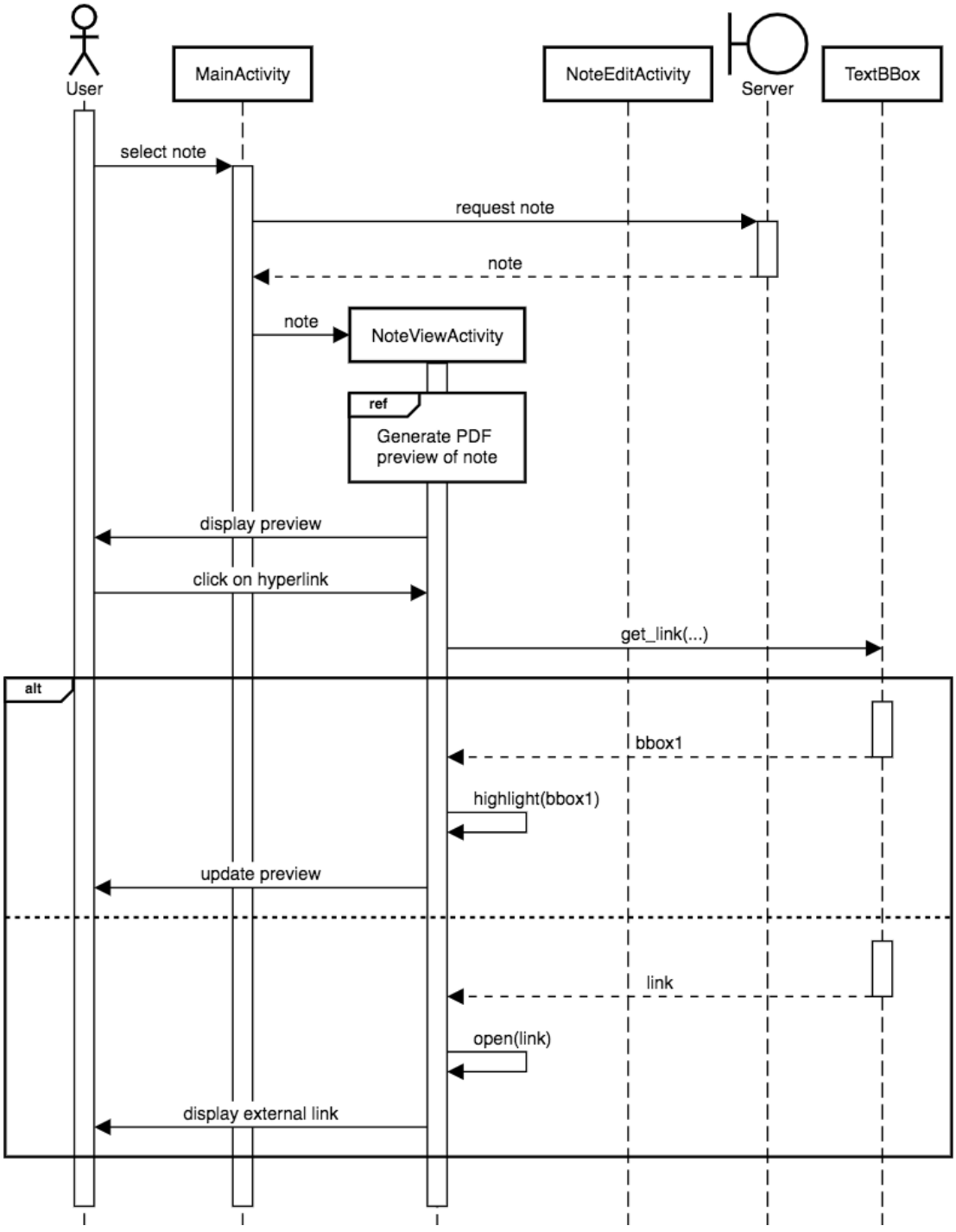

Figure 8: Viewing existing notes: the user selects a note from his main directory, the client then receives the note from the server and opens it, and the user then clicks on a hyperlink in the processed note which either links to another place in the note or loads an external webpage.

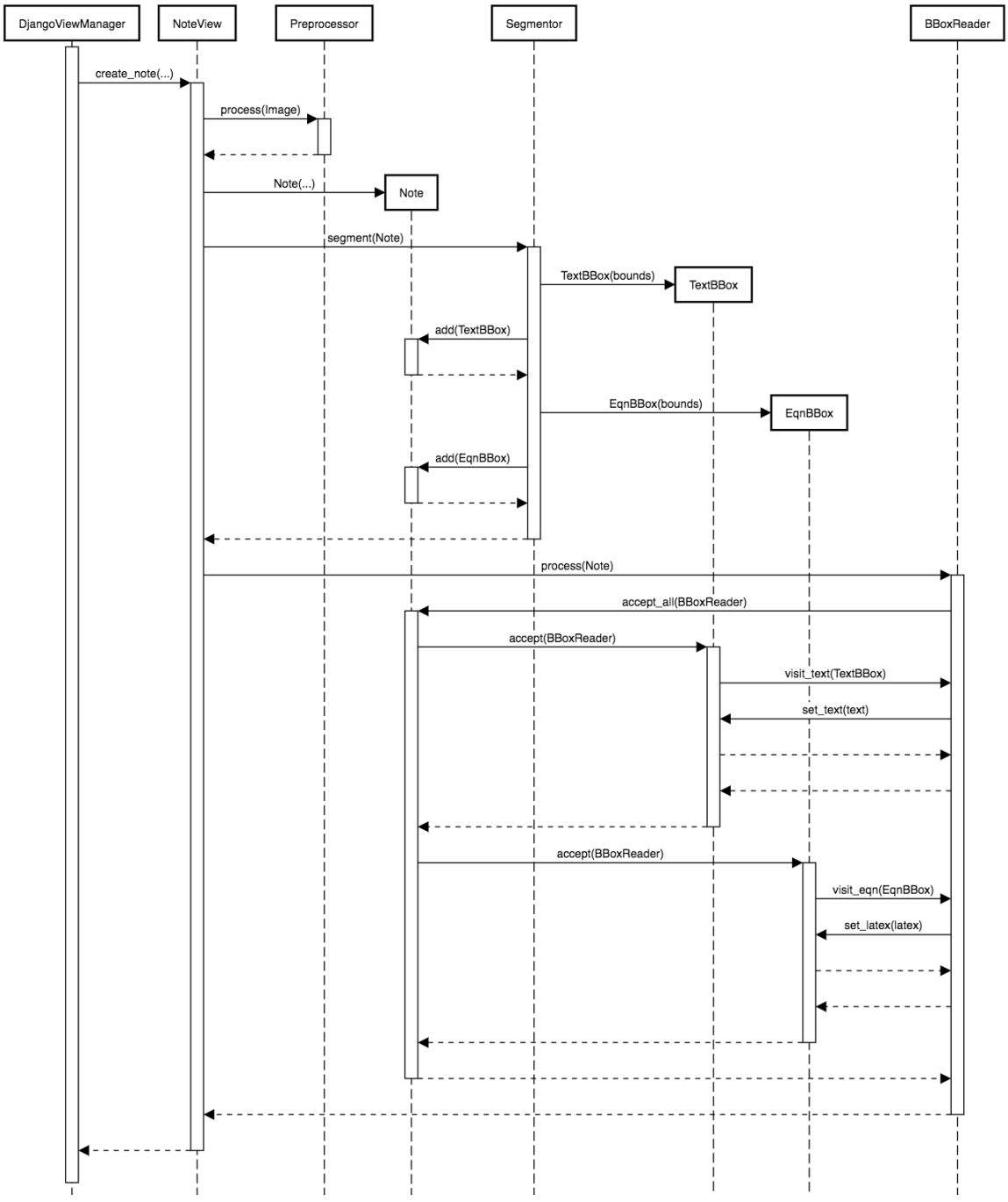

Figure 9: Sequence diagram for the creation of a new note from the server side. The image is first preprocessed (e.g. edge detection), then segmented into different regions containing text, equations, plots, figures etc., then each region is read separately.

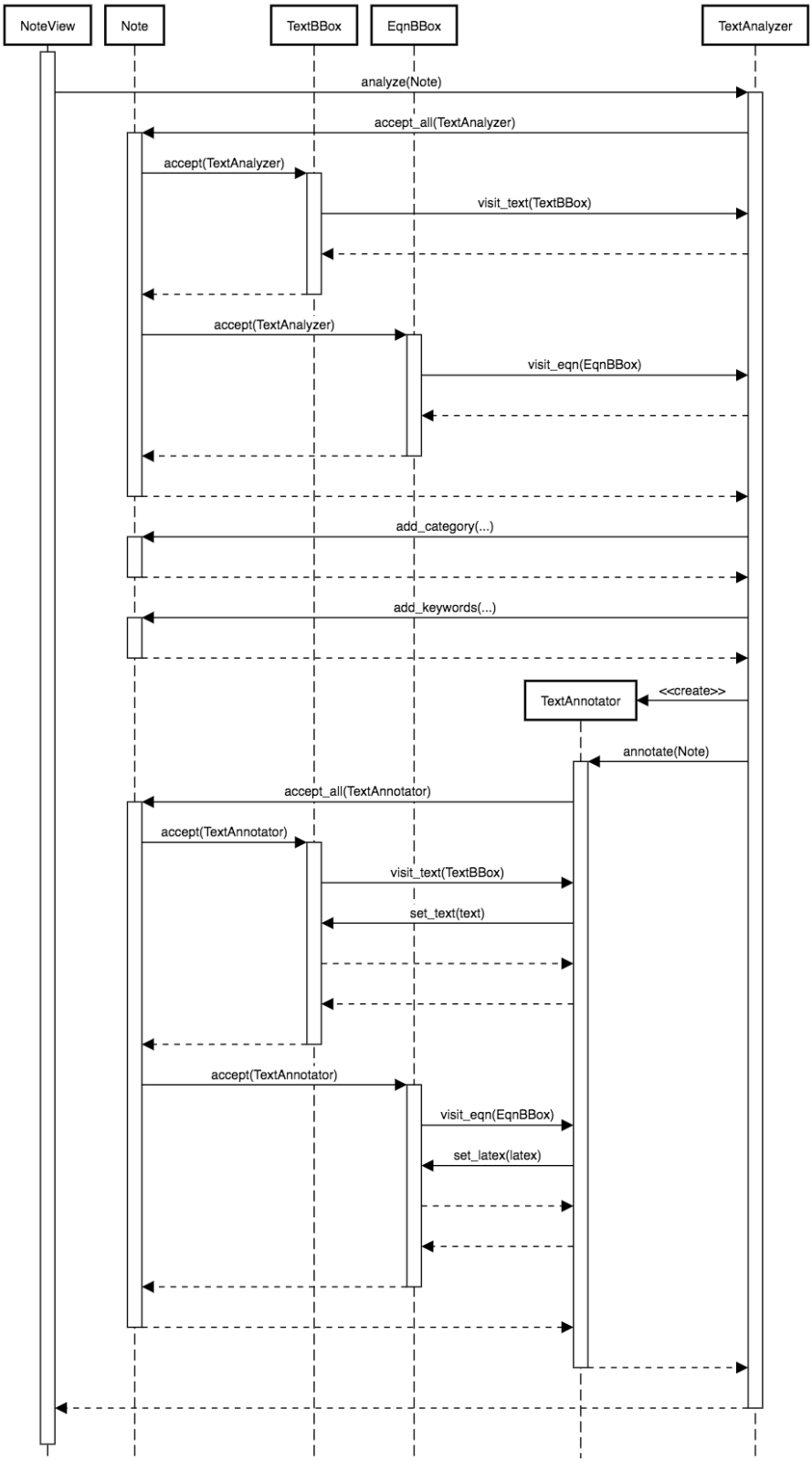

Figure 10: Sequence diagram for the annotation of a newly created note. The text and equations in the note are first analyzed to for summarization and keyword detection, and each block is then rewritten with annotations.

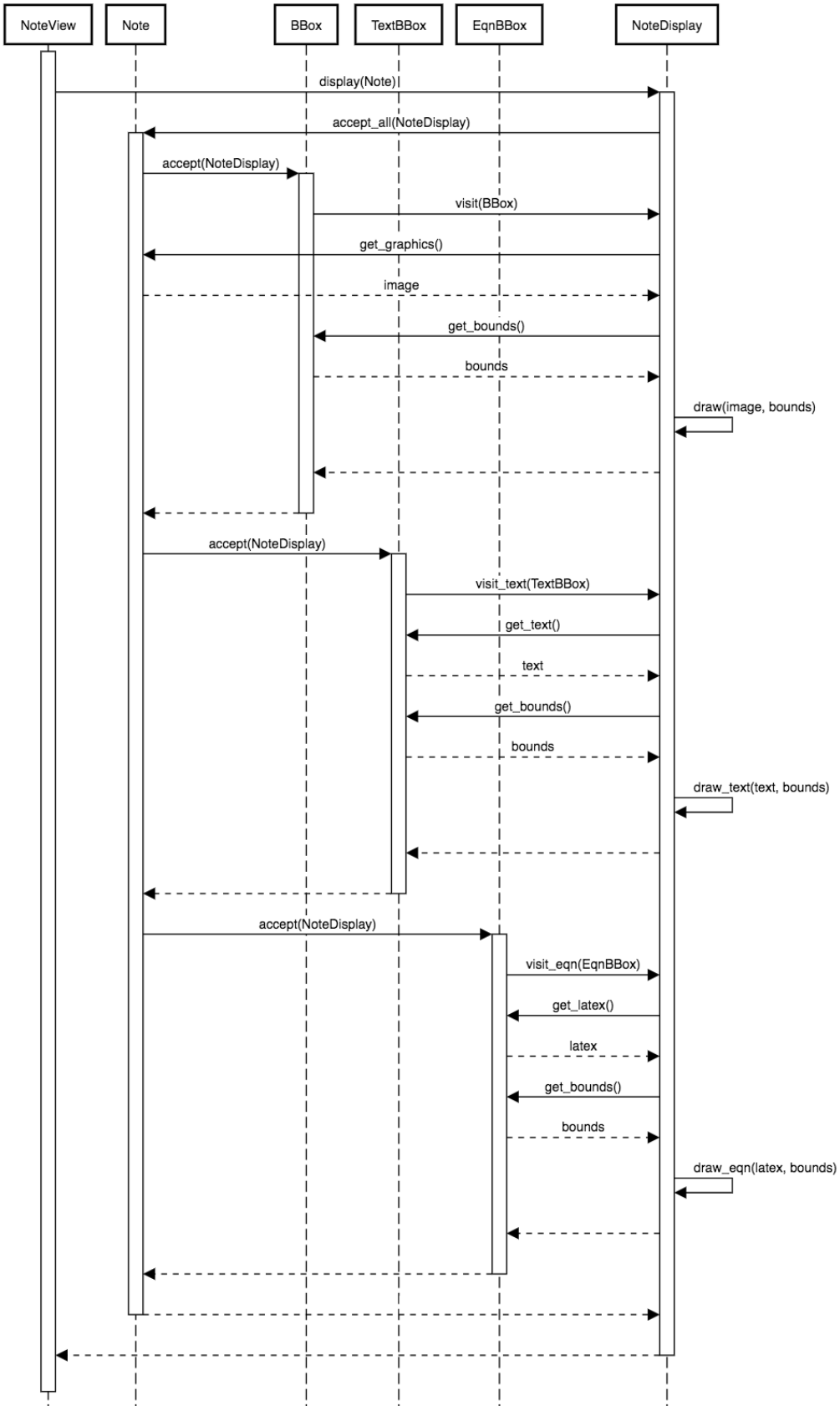

Figure 11: Sequence diagram for displaying the note visually (whether in PDF or LaTeX). Each bounding box in the note is visited and drawn in order.

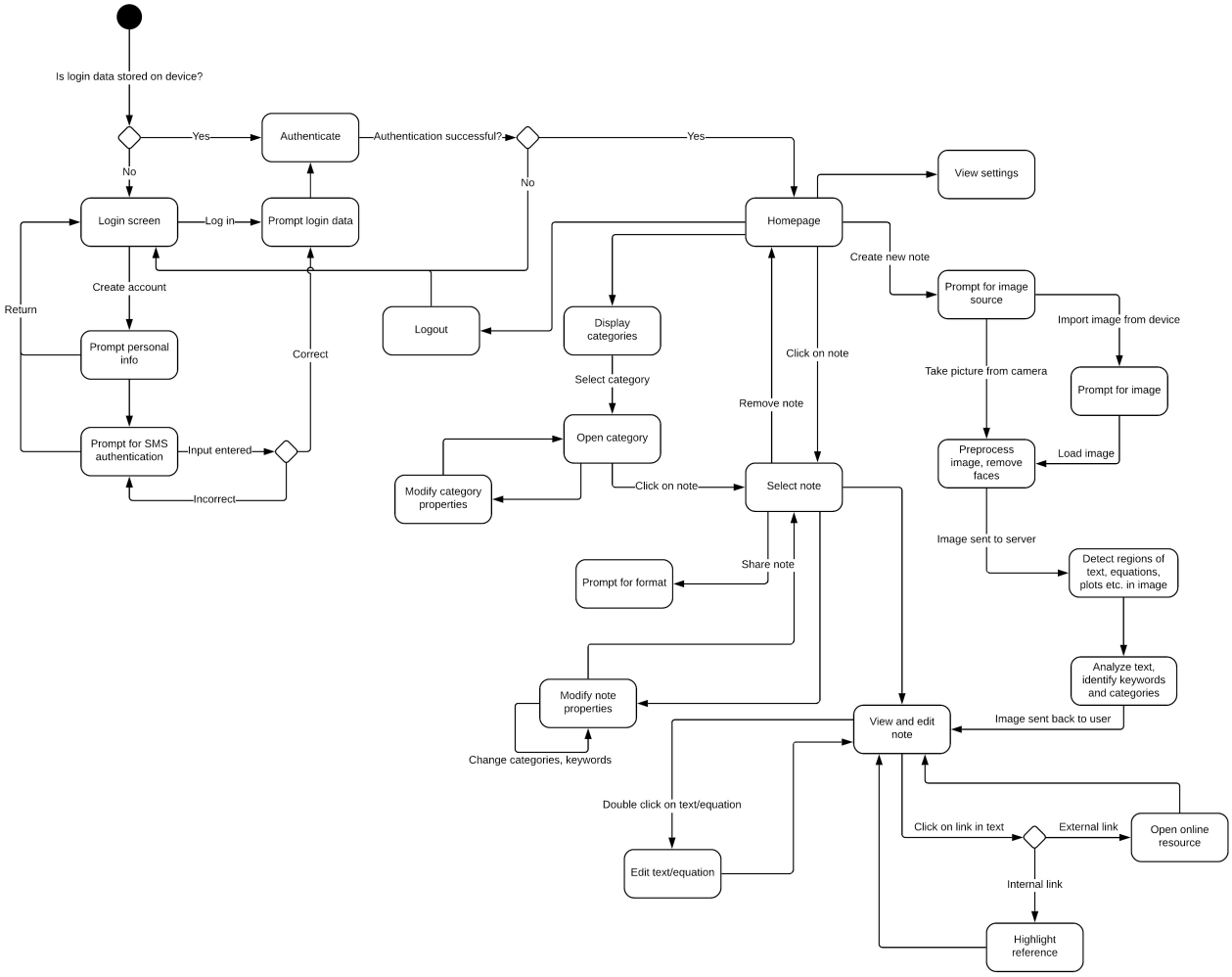

Figure 12: Activity diagram for the general control flow of the application.

2.5.5 User interface mockups

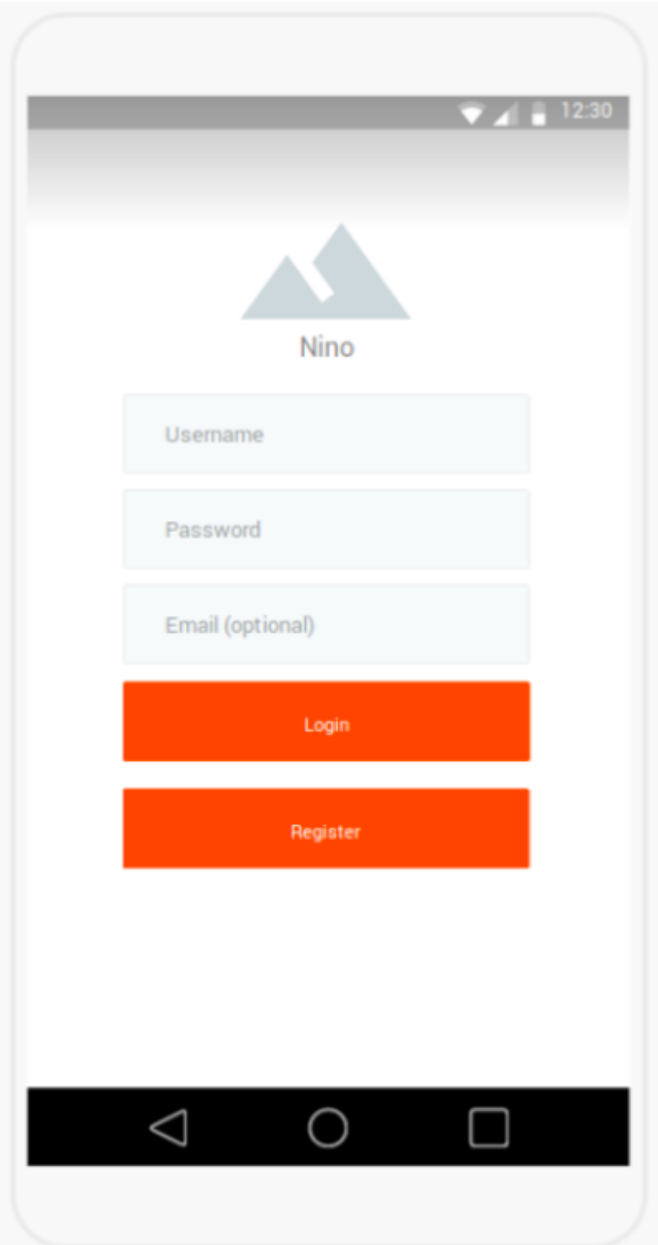

Figure 13: Register/login screen.

When the app is run, the user will be presented with an interface as the one in Figure 13. Login and register function are combined into one activity, the user will be able login and register from the same screen. Username and password are required fields for logging in and registration. The placeholder image above the text "Nino" is for Nino's future logo.

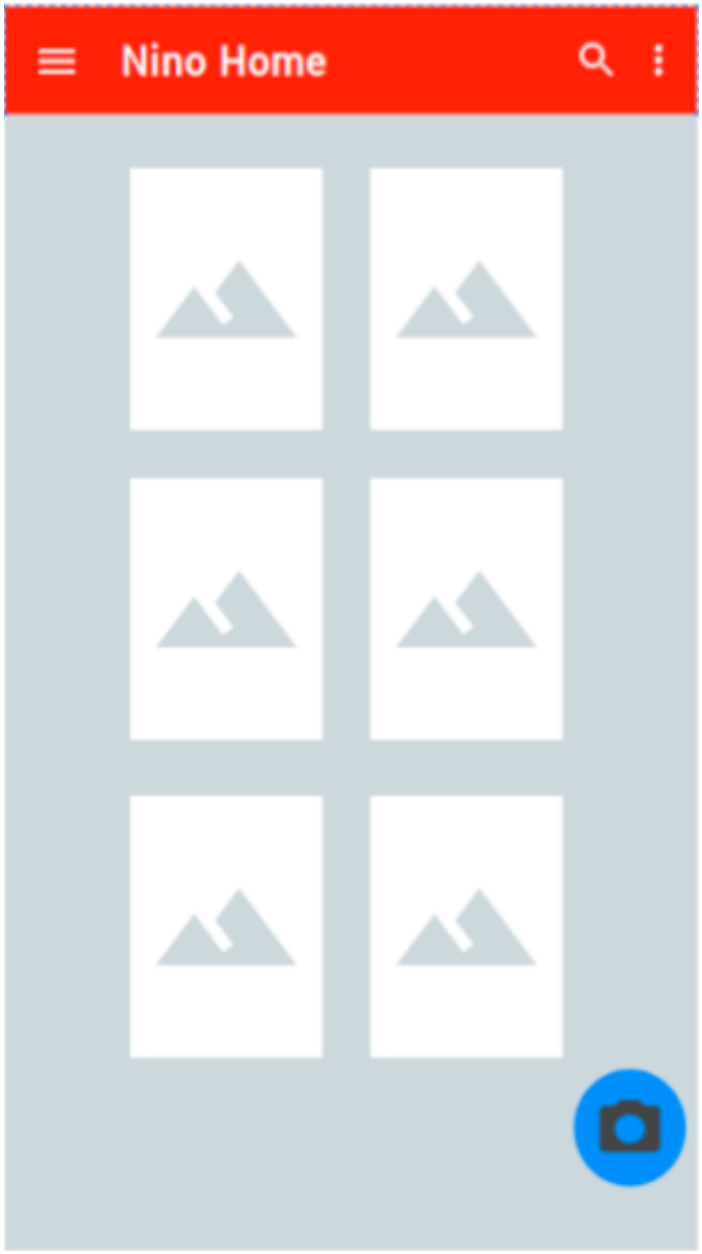

Figure 14: Home screen.

Figure 15: Three dots popup.

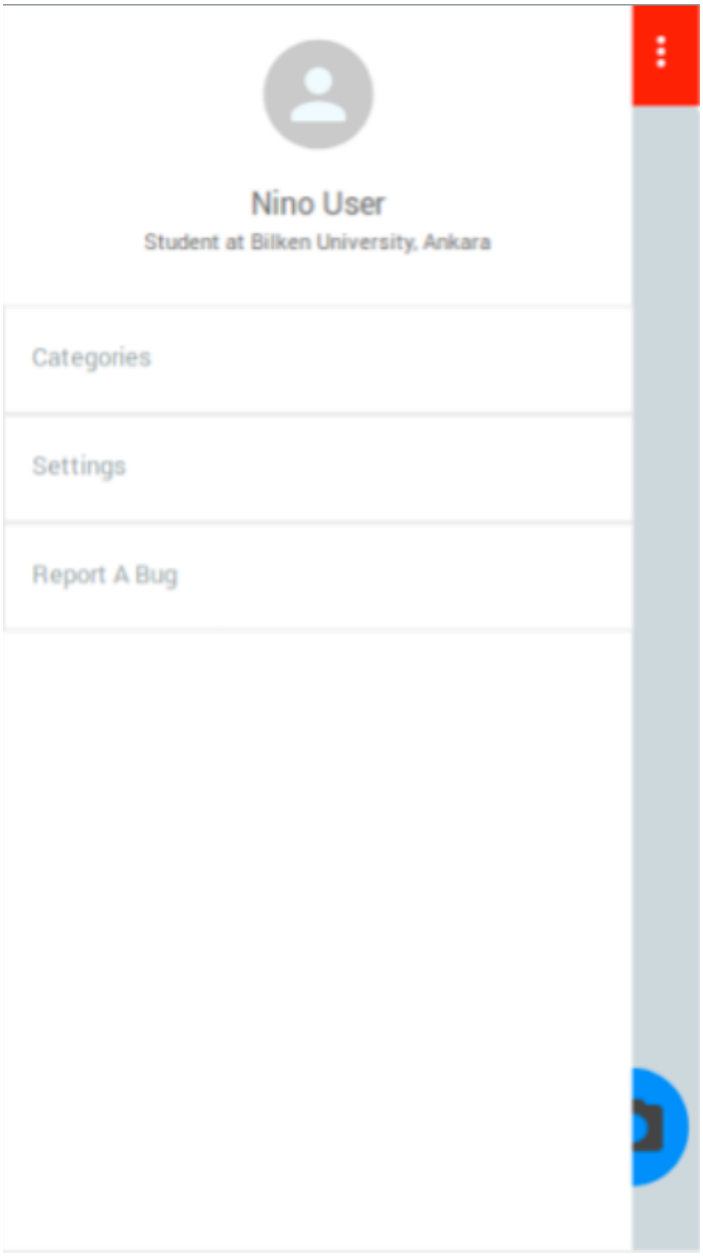

Figure 16: Navigation.

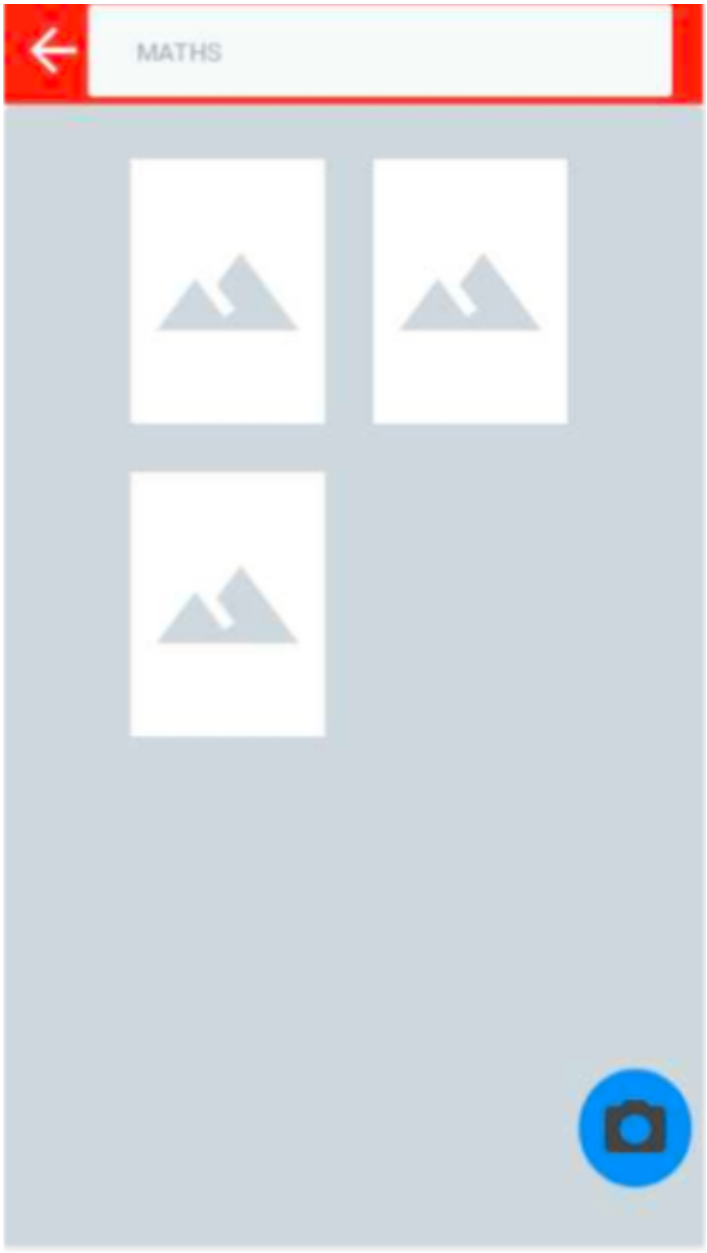

Figure 17: Search.

Figure 14 is the main screen the user is presented after logging in. On the taskbar, three dots icon on the far right opens up a list with additional options and functionality [Figure 15]. Three line icon denotes the user can open an android navigation drawer [Figure 16]. The magnifying glass icon on the right opens up a search bar [Figure 17]. Placeholder images denote the recently edited notes the user has. Clicking on the notes will take the user to the note edit screen. Clicking on the floating action button with the camera shape on bottom right will open the user's camera. The user will then take a picture and will be sent to the note edit screen afterwards, where he/she can edit the new note.

The navigation drawer in Figure 16 shows information about the user profile and functionality about the app and preferences. Figure 17 shows how the search functionality is used. Here, User searched for notes containing "MATHS", and all the notes that match the tag are shown.

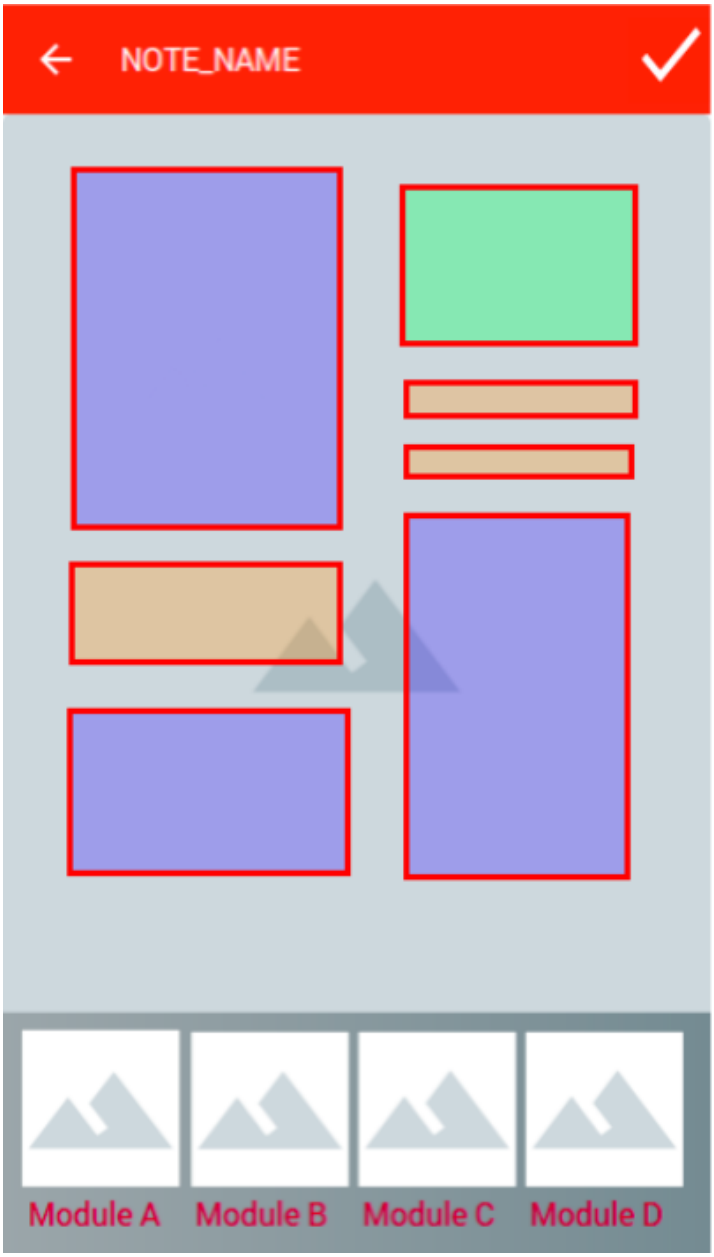

Figure 18: Note edit screen.

Figure 18 shows the note edit activity. User can navigate here either by clicking on a note on home activity, or by taking a picture using the floating action button, also located at home activity screen. The note is shown to the user in a big display. The bounding boxes shown in the figure denote different classes of regions detected in the picture by Nino. Several modules are presented on the bottom. User can select a detected region from the display to list modules that are suitable with that specific regions. For example, if we assume that the orange bounding boxes in the figure denote mathematical equations, module A might be a module that converts handwritten equations into a computer typed equation. Can select different regions and apply different modules to each region. The tick icon on top right will save any changes and send the user back to the home screen. The arrow on top left also sends the user back to the home screen, but discards any changes made to the note.

3 Glossary

Note: a handwritten picture containing regions of text, equations, plots and figures.

Sketch: any digitizable type of data apart from text and equations.

Segment: a maximal rectangular region in a note consisting of a single type of data (text, equations or sketches), enclosed by a bounding box.

Category: a directory containing notes of the same context (e.g. course topic).

References

[1] "3 Easy Steps to Digitize Your Study Notes". https://www.developgoodhabits.com/digitize-study-notes/. [Accessed Oct 15, 2018].

[2] "How to digitize your handwritten notes". https://www.popsci.com/digitize-handwritten-notes. [Accessed Oct 15, 2018].

[3] "How to Take Study Notes: 5 Effective Note Taking Methods". https://www.oxfordlearning.com/5-effective-note-taking-methods/. [Accessed Oct 15, 2018].

[4] "The LaTeX Project". https://www.latex-project.org/. [Accessed Oct 15, 2018].

[5] "PDF 2.0". https://www.iso.org/standard/63534.html. [Accessed Oct 15, 2018].

[6] "PGF and TikZ – Graphic systems for TeX". https://sourceforge.net/projects/pgf/. [Accessed Oct 15, 2018].

[7] "Creative Commons - YouTube Help". https://support.google.com/youtube/answer/2797468?hl=en. [Accessed Oct 15, 2018].

[8] "CROHME: Competition on Recognition of Online Handwritten Mathematical Expressions". https://www.isical.ac.in/~crohme/CROHME_data.html. [Accessed Oct 15, 2018].