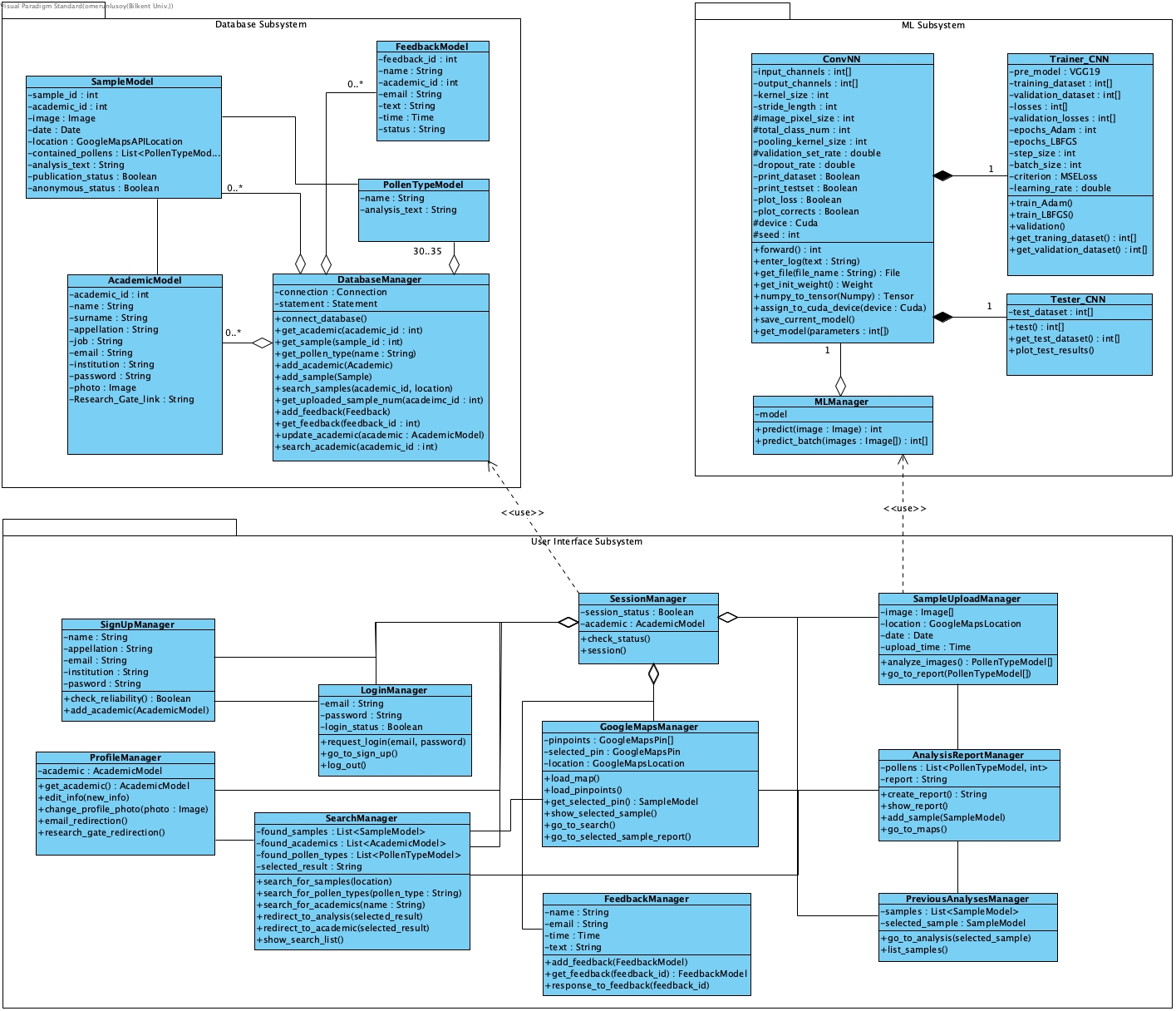
**3.5.3 Object and Class Model

3.5.3.1 ML Subsystem

3.5.3.1.1 MLManager Class

MLManager class allows PolliVidis to use our trained machine learning model with two simple functions one for single analysis and the other one for the batch analysis. This class holds the model as its instance.

3.5.3.1.2 ConvNN Class

ConvNN class is the main class of our convolutional neural network implementation. It holds several variables such as input and output channel numbers, kernel size, dropout rate, and some print control as most CNN implementations have. Moreover, it has some functions to create log files for us to track the model during its development. Finally, it has save and load model functionalities to save some time during its development process.

3.5.3.1.3 Trainer Class

Trainer class trains our CNN implementation with its two train functions, one for Adam optimizer and one for LBFGS optimizer. This class will hold the training and validation datasets we created. Several hyperparameters of the model such as step size of the optimizer and learning rate will be tuned with validation process.

3.5.3.1.4 Tester Class

Tester class will hold the test dataset we created. It will evaluate the model and calculate each evaluation matrix.

3.5.3.2 Database Subsystem

3.5.3.2.1 DatabaseManager Class

DatabaseManager class allows PolliVidis to communicate with the database of the website. This class implements all SQL query functions so that no other class has to implement queries. All User Interface Subsystem classes communicate with the database using DatabaseManager class. The class has a function to connect with the SQL database. It also has getter functions for each table. When a getter is called; DatabaseManager runs a query to get the desired table and row, then it places the returned information into the corresponding greeter model class so that the caller function can retrieve and modify the information easily without any SQL query. The class also has functions to add academic, sample, and feedback. Finally, the class has two advance search functions to search through the samples and academics.

3.5.3.2.2 PollenTypeModel Class

This model class may hold a pollen type when the DatabaseManager class queries its corresponding SQL table. While the number of pollen types our model will use does not change on the SQL table, active model class number can change during a session.

3.5.3.2.3 AcademicModel Class

This model class may hold an academic account when the DatabaseManager queries its corresponding SQL table. The class may hold all of the academic profile related variables such as name, job, institution etc. This model class can be modified as the academic edits his/her profile page. Moreover, this model class will be used when a user tries to access the communication information of an academic.

3.5.3.2.4 SampleModel Class

SampleModel class may hold a sample image with its analysis report when the DatabaseManager retrives a sample from the database. This model will be used when a user tries to view a sample or when an academic wants to upload his/her sample with its analysis to our database.

3.5.3.2.5 FeedbackModel Class

This model class holds feedbacks before they push to the database.

3.5.3.3 User Interface Subsystem

3.5.3.3.1 SessionManager Class

SessionManager class handles each session created by a user. As PolliVidis is a website rather than an application, its backend will be codded as session-based. When a user enters PolliVidis website, SessionManager will create a session for him/her. It holds the session information with the logged in academic.

3.5.3.3.2 SampleUploadManager Class

SampleUploadManager class handles the sample upload of a user when s/he wants to analyse a sample. A sample model will be created and the image(s) will be sent to the MLManager. When the MLManager returns with the result, this class will direct user to the Analysis Report page to view the report.

3.5.3.3.3 AnalysisReportManager Class

This class handles the analysis report of a sample. When the user analyzes a sample or wants to view an analysis of someone else, this class generates or retrieves the report. It also uploads the sample object with its report to the database using DatabaseManager.

3.5.3.3.4 PreviousAnalsesMangerClass

This manager class retrieves all previous analyses of an academic and lists them when s/he wants to review them. It can redirect academic to the Analysis Report page when s/he wants to review one of the analyses.

3.5.3.3.5 GoogleMapsManager Class

This manager handles Google Maps API and Maps page, it allows users to use pinpoints and basic map functionalities. When the user wants to review a sample by clicking a pinpoint, this manager handles the sample retrieval and Sample Info panel.

3.5.3.3.6 SearchManager Class

SearchManager class handles sample, pollen type, and academic info searches with the help of the DatabaseManager class’s search functions. It lists all the related results within the Search panel.

3.5.3.3.7 LoginManager Class

This class handles academic logins to the website. It checks the database for the given e-mail and password. It also ends the session when the academic logs out.

3.5.3.3.8 SignUpManager Class

SignUpManager handles the academic sign-ups to PolliVidis with proper reliability check as PolliVidis is allows academic accounts only. When the academic signs up, this class adds his/her model object to the database.

3.5.3.3.9 ProfileManager Class

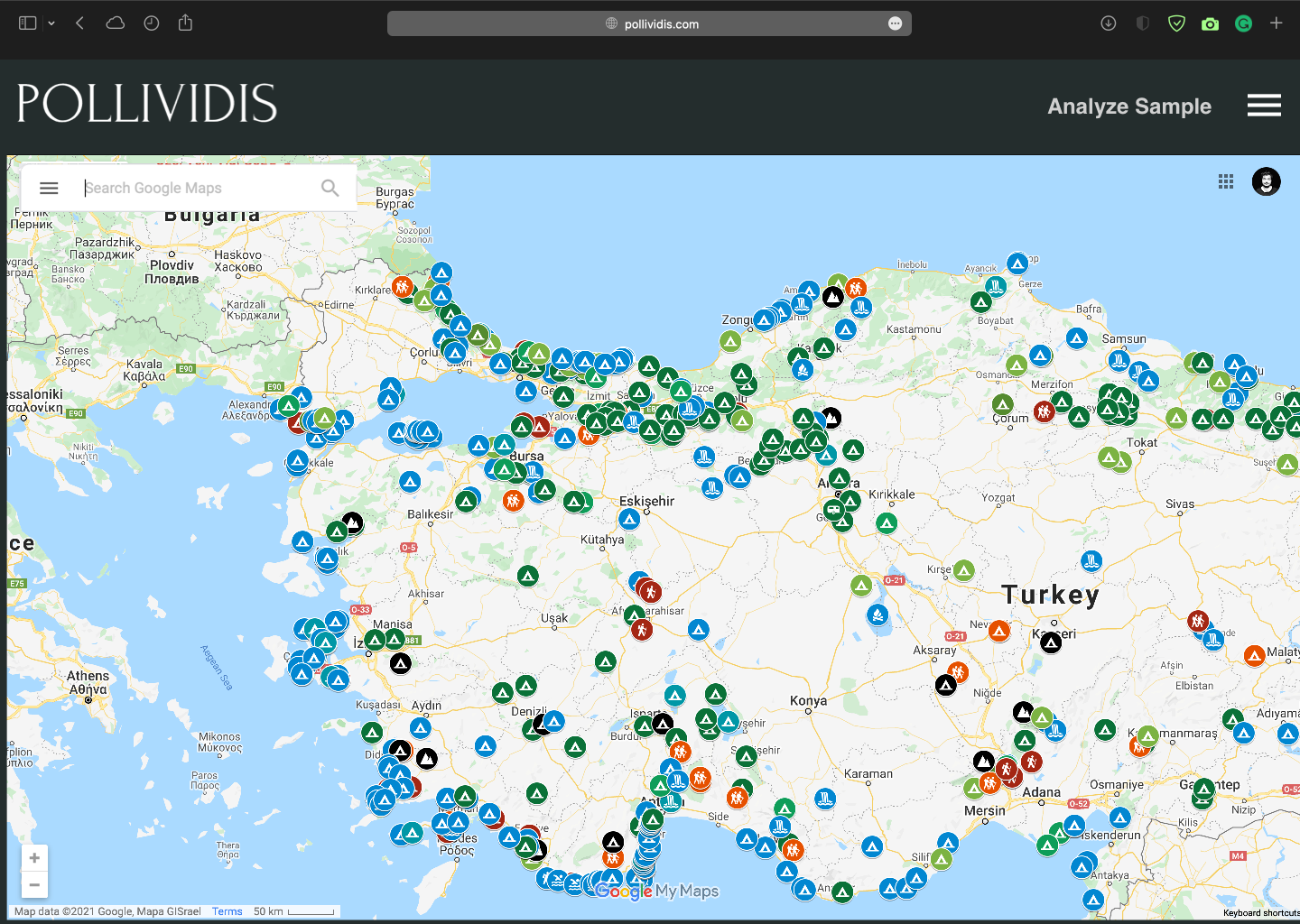
This manager class allows academics to view and edit their profiles. Moreover, it authorizes the users who want to view the communication information of the academic.

3.5.3.3.10 FeedbackManager Class

FeedbackManager class allows all users to send feedback to us and get help when it is needed. It creates a feedback model object and puts it into the database using DatabaseManager. As it also takes users communication information, we can return to the user with proper answer. Finally, any academic can ask us questions about the website and the model using Feedback page.

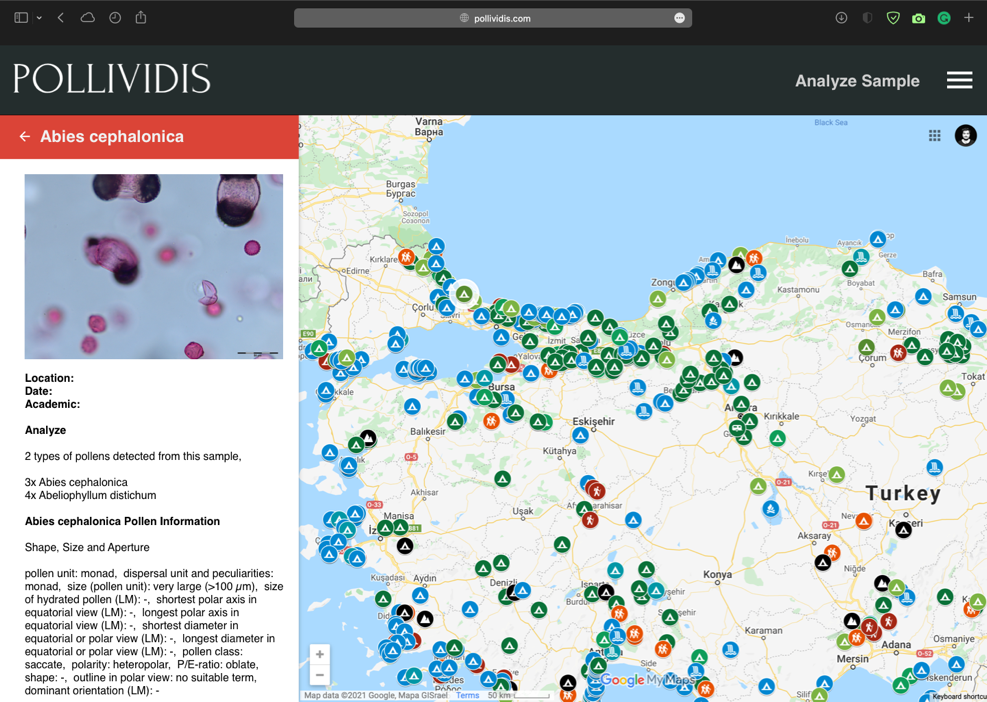
3.5.5 User Interface

3.5.5.1 Map Page



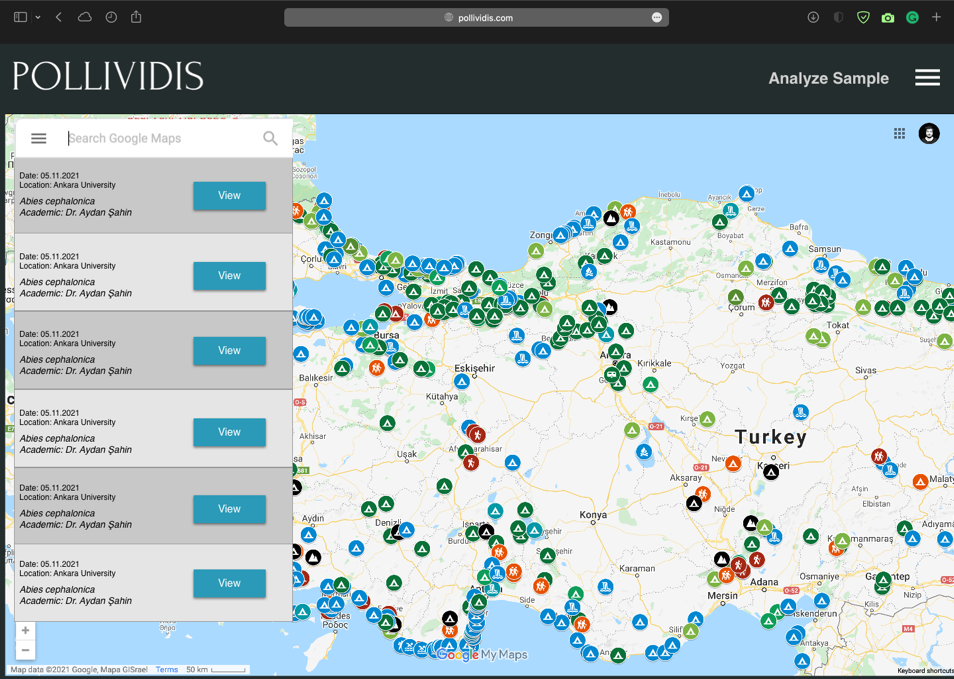
Map page of PolliVidis is the first page a user sees. As PolliVidis supports anonymous usage for viewing analyses and analyzing samples, it welcomes users with its Google Maps integrated map page rather than a login screen. Each pinpoint on the map represents a sample analysis and any anonymous user can view them by clicking on them. The map supports basic functionalities of Google Maps such as zoom in and zoom out. A user can go to the Analyze Sample (Upload) Page by just clicking on “Analyze Sample” button. Moreover, the options panel can be opened with “options button” (three lines). Finally, a user can use all PolliVidis search functionalities with the search bar.

3.5.5.2 Map Page with Sample Info Panel



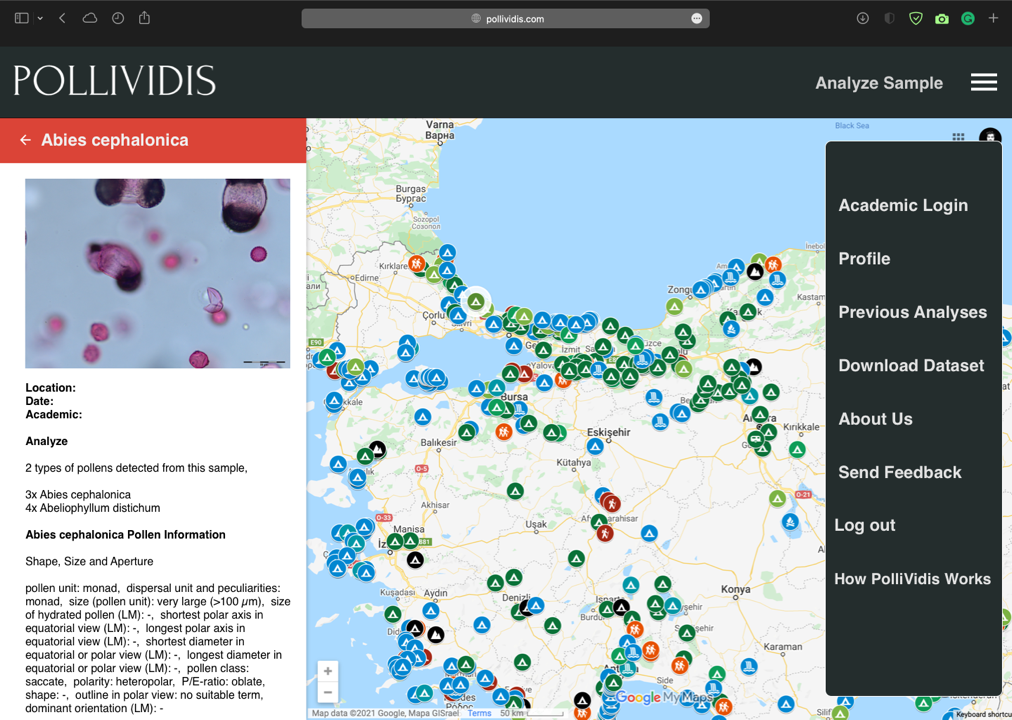
When a user clicks on a sample (pinpoint) on the map, Sample Info Panel opens at the right side. This panel shows the sample image with its analysis report. This panel allows users to scan the map fast by switching between samples. If the user wants to see the detailed analysis report, s/he can go to the sample’s analysis report by clicking on the sample image.

3.5.5.3 Search Page



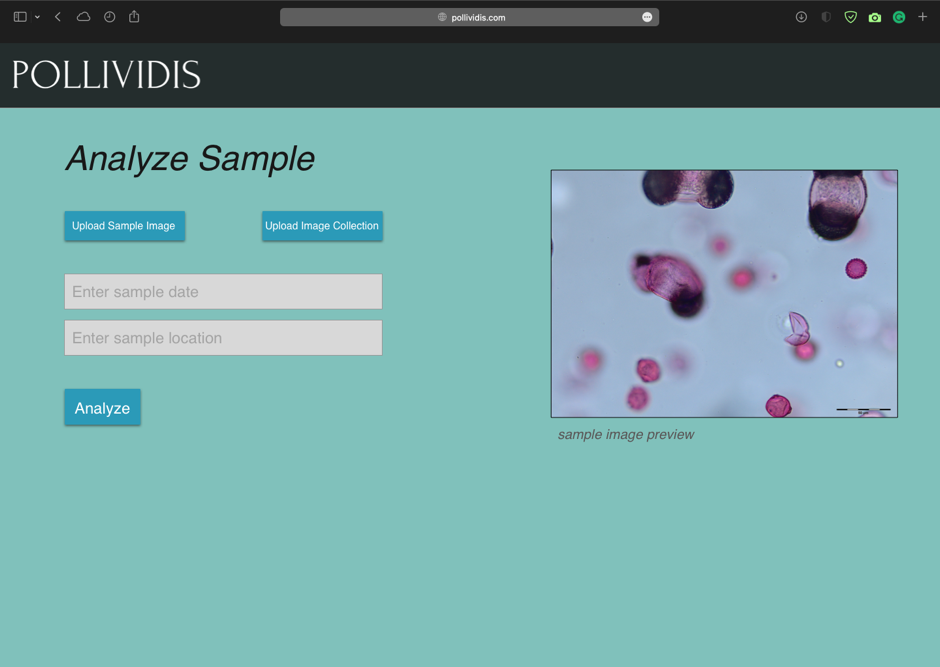
Search bar at the top right corner of the Map Page allows users to use all PolliVidis search functionalities. These search functionalities include searching an academic (for academic accounts only), a location (such as city or building), or a pollen species. These search functionalities relieve user from wandering on the map.

3.5.5.4 Options Panel Page



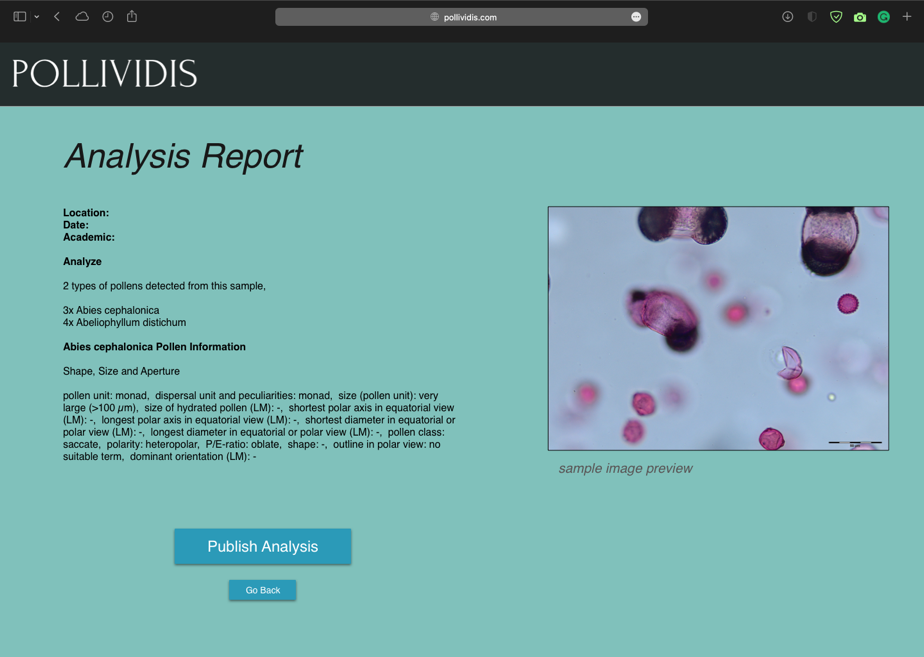
Options panel allows users to login, sign-up, go to their profiles (academic users only), view their previous analyses, send feedback to us, log out, and download the dataset. Although, different pages and users will see different options in the panel, we have included all the options in the mockup to avoid making several mockups just for this page.

3.5.5.5 Analyze Sample (Upload) Page

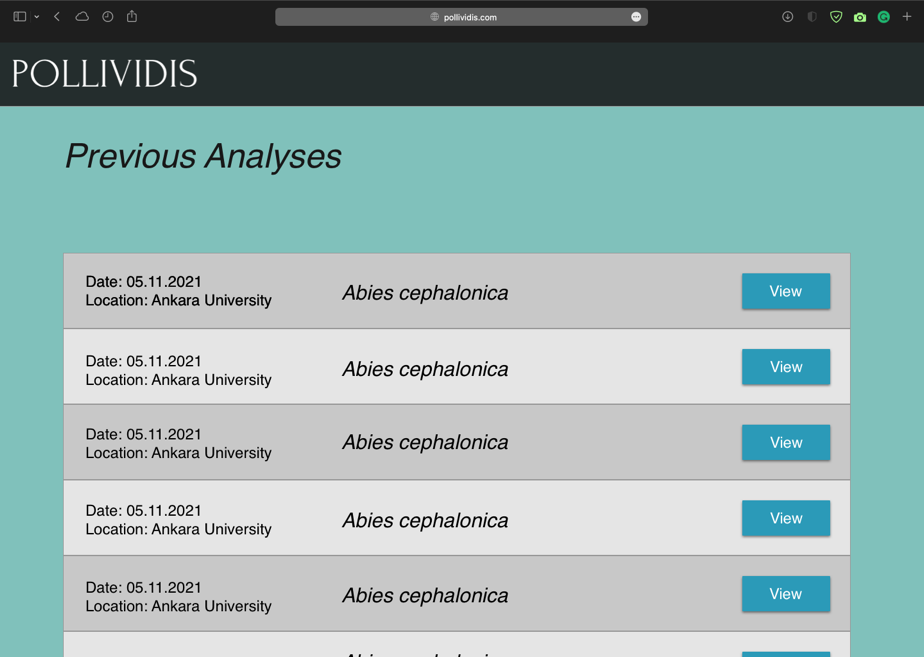


Analyzing a sample with PolliVidis is available for all users and this page can be accessed directly from the Map Page. A user can upload one sample image to analyze or image collection for collective analysis. The page asks academic users for the date and the location of the sample to upload the sample analysis to the database if they allow it.

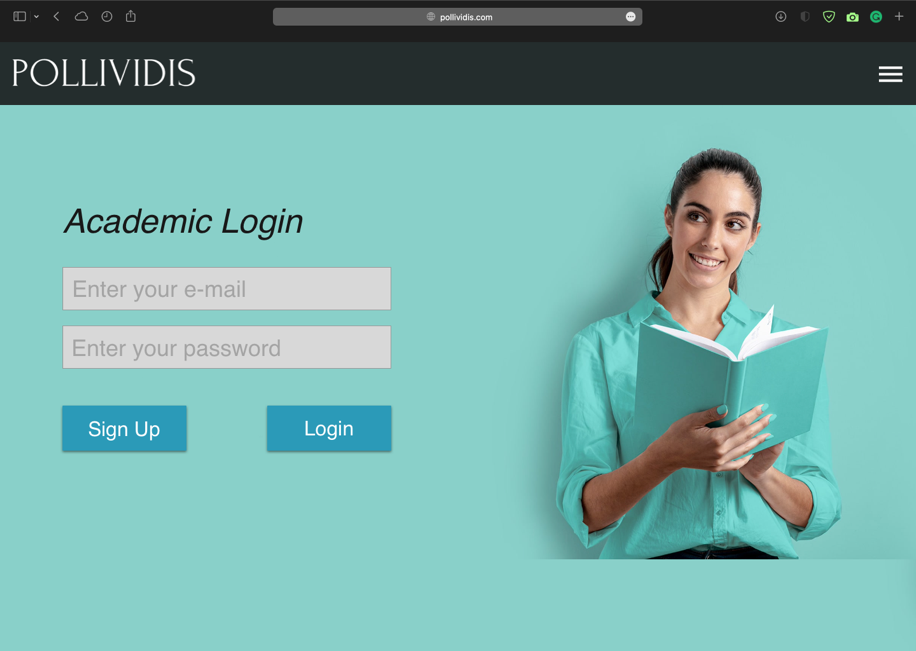
3.5.5.6 Analysis Report Page



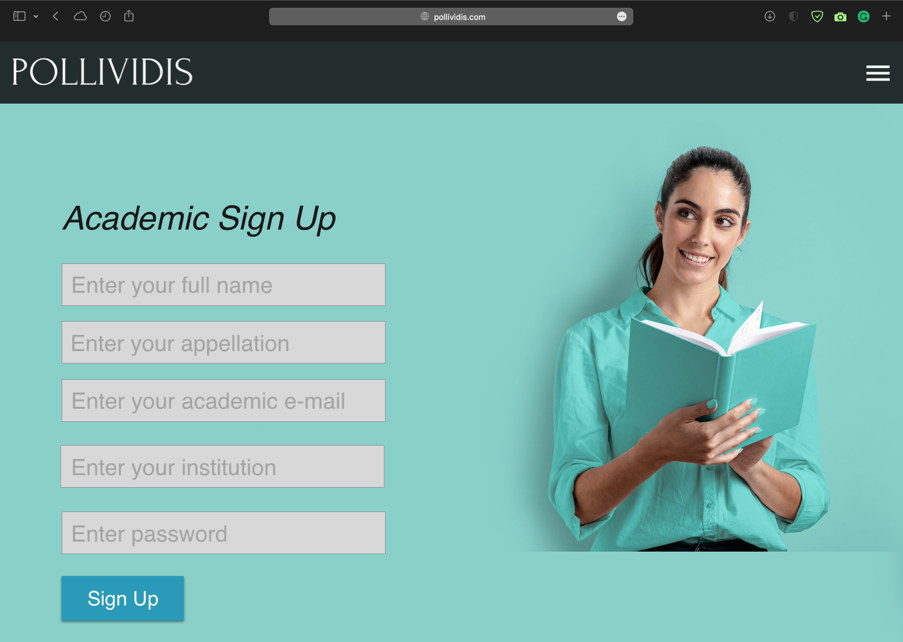
After uploading a sample image and clicking on the “analyze button”, all users are redirected to the Analysis Report Page to view the detailed analysis of their samples. This report includes the location, date, academic information, pollen species, and detailed analysis of each species of the given sample. If the user analyzing the sample is an academic, this page asks him/her for a permission to upload the analysis to the database. Moreover, any user will be directed to this page when they want to view the detailed analysis of any sample.

3.5.5.7 Previous Analyses Page

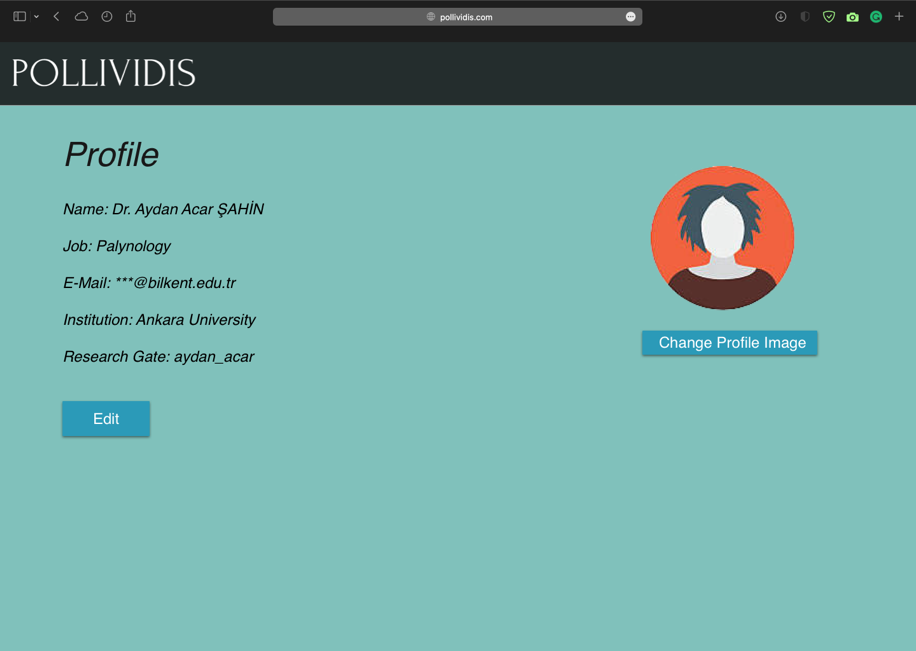
Previous Analyses Page lists all the previous samples and their analyses of the academic. Since we will not store any information for anonymous sample analyses, this page can only be used by academic accounts. However, anyone can view previous analyses of any academic with their Previous Analyses Page if the academic allows it.

3.5.5.8 Academic Login Page

Academic Login Page allows academics to login their accounts for further advantages of an academic account such as Previous Analyses Page and detailed communication information of other academics. Any academic without an account can go the Sign-Up Page using “Sign Up button”.

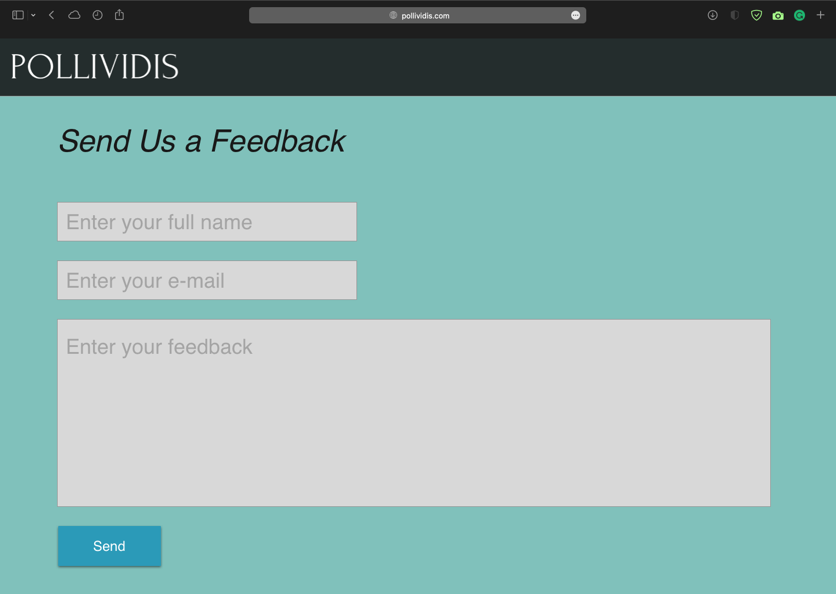
3.5.5.9 Academic Sign-Up Page

Any academic can create a PolliVidis account by supplying required information such as appellation, e-mail, and institution to benefit from further advantages of an academic account. A verification procedure will be used to ensure the reliability of the supplied information.

3.5.5.10 Academic Profile Page

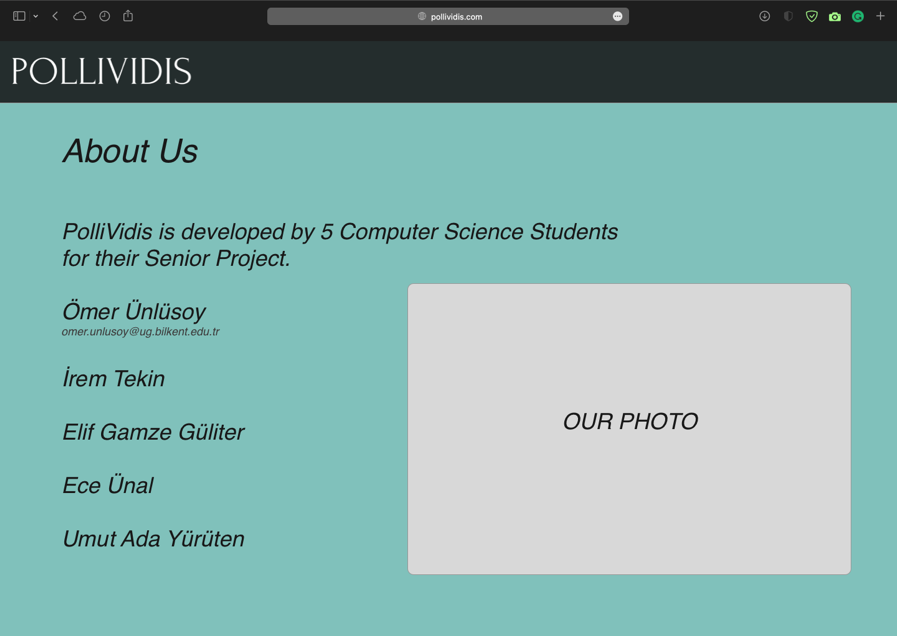
Each academic account can view their profile and edit the supplied information using Academic Profile Page. This page also includes the communication information and a photo of the academic.

3.5.5.11 Send Feedback Page



Any user can send us feedback using Send Feedback Page. Full name and an e-mail account should be supplied so that we can respond.

3.5.5.12 About Us Page



Some information about each team member with a group photo can be viewed at About Us Page.

3.5.5.13 How It Works Page

As an extra page, How It Works Page explains the Machine Learning model we will use and the analysis procedure for curious users. Since the page will just include some explanation about the model and a representative image of the model’s layer which neither of them ready, we did not design a mockup for this page.

## Project Plan

|  |  |  |  |
| --- | --- | --- | --- |
| WP# | Work Package Title | Leader | Members Involved |
| WP1 | Preparing Pollen Dataset | İrem Tekin | Ömer Ünlüsoy |
| WP2 | Preparing the Pollen Extraction Algorithm | Umut Ada Yürüten | İrem Tekin |
| WP3 | Transfer Learning Process | Ömer Ünlüsoy | Elif Gamze Güliter |
| WP4 | Data Augmentation Process | Elif Gamze Güliter | İrem Tekin  Ece Ünal  Ömer Ünlüsoy |
| WP5 | Preparing the First CNN Model | Ömer Ünlüsoy | Umut Ada Yürüten  Ece Ünal  İrem Tekin  Elif Gamze Güliter |
| WP6 | Backend of the Pollividis Website | Ece Ünal | İrem Tekin  Ömer Ünlüsoy |
| WP7 | Finalizing the CNN Model | Ömer Ünlüsoy | Ece Ünal  Elif Gamze Güliter  Umut Ada Yürüten  İrem Tekin |
| WP8 | Front-end Implementation and Integration with Google Maps API | İrem Tekin | Ece Ünal  Umut Ada Yürüten |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP1:** Preparing Pollen Dataset | | | |
| Start Date: 4 October 2021 | | **End Date:** 24 December 2021 | |
| Leader: | İrem | Members Involved: | Ömer |
| **Objectives:** 30-35 pollen species will be collected. 200-500 photographs will be taken for each species. It will serve for both our model and feature researches. | | | |
| Tasks:  **Task 1.1:** Photograph each species at Ankara University Palynology Laboratory. | | | |
| Deliverables:  **D 1.1:** Dataset consisting of30-35 pollen species with 200-500 photos | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP2:** Preparing the Pollen Extraction Algorithm | | | |
| **Start Date:** 25 October 2021 | | **End Date:** 25 November 2021 | |
| Leader: | Umut Ada | Members Involved: | İrem |
| **Objectives:** To implement a pollen extraction algorithm to extract each pollen photo separately from a given sample. | | | |
| Tasks:  **Task 2.1:** Search the existing extraction algorithms.  **Task 2.2:** Implement a pollen extraction algorithm in Python.  **Task 2.3:** Implement a main class for pre-training process which will take samples from file directories, prepare them for the model, and call the model’s predict function. | | | |
| Deliverables:  **D 2.1:** Pollen extraction algorithm to extract pollens from samples before CNN  **D 2.2:** A main class for pre-training process | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP3:** Transfer Learning Process | | | |
| **Start Date:** 25 October 2021 | | **End Date:** 14 March 2022 | |
| Leader: | Ömer | Members Involved: | Elif Gamze |
| **Objectives:** Using a pre-trained model to increase our model’s accuracy due to the shortage of our dataset. | | | |
| Tasks:  **Task 3.1:** Search available pre-trained models (like VGG-19), and decide the one to use in our model.  **Task 3.2:** Implement the first version of transfer learning for our model in PyTorch.  **Task 3.3:** Run several tests to evaluate the compatibility of the pre-trained model.  **Task 3.4:** Adjust the hyperparameters of the model to increase accuracy.  **Task 3.5:** Finalize the pre-trained model with small adjustments. | | | |
| Deliverables:  **D 3.1:** Pre-trained model with hyperparameter adjustment | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP4:** Data Augmentation Process | | | |
| **Start Date:** 25 October 2021 | | **End Date:** 27 November 2021 | |
| Leader: | Elif Gamze | Members Involved: | İrem, Ece, Ömer |
| **Objectives:** Finding the applicable Data Augmentation methods for our model to increase the dataset size and decrease overfitting. | | | |
| Tasks:  **Task 4.1:** Search the Data Augmentation methods and decide which ones are suitable for our model and dataset.  **Task 4.2:** Implement the data augmentation (transformation) in PyTorch. | | | |
| Deliverables:  **D 4.1:** Data Augmentation (transformation in PyTorch) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP5:** Preparing the First CNN Model | | | |
| **Start Date:** 15 November 2021 | | **End Date:** 15 December 2021 | |
| Leader: | Ömer | Members Involved: | Umut Ada, Ece, İrem, Elif Gamze |
| **Objectives:** Implement the first version of Pollen Classifier CNN model. | | | |
| Tasks:  **Task 5.1:** Each team member will learn the neural network and PyTorch basics.  **Task 5.2:** Search similar CNN model implementations to get an idea about the required structure of the model.  **Task 5.3:** Implement the first version of convolutional, max pooling, and fully connected layers with PyTorch Sequential.  **Task 5.4:** Implement Trainer class.  **Task 5.5:** Implement Tester class. | | | |
| Deliverables:  D 5.1: CNN class  **D 5.2:** Trainer class  **D 5.3:** Tester class | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP6:** Backend of the Pollividis Website | | | |
| **Start Date:** 15 October 2021 | | **End Date:** 14 March 2022 | |
| Leader: | Ece | Members Involved: | İrem, Ömer |
| **Objectives:** Implement the project backend for database, ML model, and UI connections in Python Django. | | | |
| Tasks:  **Task 6.1:** Each involved team member will learn the basics of Python Django framework which we will use to implement PolliVidis website backend.  **Task 6.2:** Implement the database with SQL.  **Task 6.3:** Connect the Django backend of the project with the initialized database.  **Task 6.4:** Implement query functions which will upload and request samples from the datasets.  **Task 6.5:** Connect the backend with the ML model.  **Task 6.6:** Implement the function that will produce the analysis report using ML model classification.  **Task 6.7:** Connect the backend with the user interface of PolliVidis. | | | |
| Deliverables:  **D 6.1:** Database Implementation with SQL  **D 6.2:** PolliVidis Backend | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP7:** Finalizing the CNN Model | | | |
| **Start Date:** 15 December 2021 | | **End Date:** 10 April 2022 | |
| Leader: | Ömer | Members Involved: | Ece, Elif Gamze, Umut Ada, İrem |
| **Objectives:** To finalize the implemented CNN model with proper tests, hyperparameter adjustments, and several optimizations. | | | |
| Tasks:  **Task 7.1:** Implement different versions of the first CNN with different pre-trained models.  **Task 7.2:** For each implementation, run several tests and adjust hyperparameters to increment the evaluation matrices including accuracy.  **Task 7.3:** Compare each implementation and decide the final structure of the model. | | | |
| Deliverables:  **D 7.1:** CNN model | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **WP8:** Front-end Implementation and Integration with Google Maps API | | | |
| **Start Date:** 15 November 2021 | | **End Date:** 10 April 2022 | |
| Leader: | İrem | Members Involved: | Ece, Umut Ada |
| **Objectives:** To implement the user interface of PolliVidis. | | | |
| Tasks:  **Task 8.1:** Each involved team member will learn HTML, React basics.  **Task 8.2:** Implementation of each mockup.  **Task 8.3:** Integration of Google Maps API.  **Task 8.4:** Final adjustments between user interface, backend, and database. | | | |
| Deliverables:  **D 8.1:** User Interface of each PolliVidis page | | | |