Time Series Forecasting & Benchmarks Repository Software Design Specification

Aiden Duval, Zane Globus-O'Harra, Erin Stone, Kareem Taha, Zachary Weisenbloom
Team 1 ("zeakz")
https://github.com/zfgo/CS_422_p1

CS 422 Spring 2023 Project 1 – 2023-05-05

Table of Contents

1 Revision History

Date	Author	Description
2023-04-13	Aiden D, Kareem T	Add initial document to Google Drive, add table of contents, and add section and subsection titles. Start 2.
2023-04-16	Zane G-O	Finish 2. Add 3.1, 3.2, and 3.3.
2023-04-19	Zane G-O	Update 2 to include information about MVC/MTV. Add 4, 4.1 (including all subsubsections), 4.2 (including all subsubsections), and 4.3 (including all subsubsections).
2023-04-20	Kareem T	Update 2, document reformatting.
2023-04-21	Zane G-O	Add 4.4 (including all subsubsections), 4.5 (including all subsubsections), and 4.6 (including all subsubsections).
2023-04-23	Zane G-O	Add 4.7 (including all subsubsections), and 4.8 (including all subsubsections). Add 5.1, 5.2, 5.3, 5.4, 6, and 7.
2023-05-04	Zane G-O, Erin S	Small format changes. Update 2, 3.1, 3.2, 3.3, 4.1.1, 4.1.2, 4.1.3, 4.1.4, 4.1.5, 4.1.6, 4.2.1, 4.2.2, 4.2.3.1, 4.2.5, 4.2.6, 4.3.1, 4.3.2, 4.3.3, 4.3.4, 4.3.5, 4.3.6, 4.5.1, 4.5.2, 4.5.3, 4.5.4, 4.5.6, 4.6.1, 4.6.2, 4.6.3, 4.6.4, 4.6.5, 4.6.6, 4.7.1, 4.7.2, and 4.7.5 to meet final product's software design.
2023-05-05	Zane G-O	Update 4.4, 4.8 to reflect the fact that we were unable to integrate the comparison metrics functionality.

2 System Overview

A time series (TS) measures a variable's change over time, and creating forecasts of these series is utilized in a variety of industries to produce predictions and forecasts, as well as analyze, and assess current trends. Financial portfolio evaluations, weather monitoring, economic trends, and stock analysis comprise a few of the many applications of TS forecasting. As such, developing models and forecasts capable of properly predicting a given quantity's future value is incentivized by the value of information gained and/or the potential to extract profit.

Machine learning (ML) is a field composed of various techniques to model statistical trends; as the technology around ML has evolved, so has the growth and application of ML techniques in academic settings and industry settings. Data scientists (DS) and ML engineers (known as "participants") create forecasting algorithms or build ML models that can be used with TS data to create predictions for what might occur with that data in the future.

Our project serves the purpose of connecting users (known as "contributors") who upload TS tasks, TS test data, and TS train data, with participants, who are capable of building a functional ML model or forecasting algorithm from the data. These models and algorithms produce a set of solution data based on the forecasting task uploaded by the contributor. The solution data is delivered back to our system, where comparison metrics, statistics, and graphs are produced comparing the participant's solution data to the test data uploaded by the contributor.

Our system acts as a TS repository, where TS data can be uploaded by contributors, and downloaded by participants, who will produce a solution and use our system to assess their solution. This system consists of six main components:

~~1. User account information.~~
~~2. A login page.~~ A home page for users to choose to be a contributor or participant.
3. Forecasting task information and TS data sets.
4. A page for contributors to upload TS data (test and train sets).
5. A page for participants to download TS data (train sets only).
6. A page for participants to upload their solution and receive feedback in the form of comparison metrics.

The ~~user account information and login~~ home page will be used to ~~authenticate users, as well as~~ provide users with the choice of registering as a contributor or as a participant. ~~The login page also offers users a way of accessing their previously created account.~~ The forecasting task information and TS data sets are stored in a database by the system, where the train data sets can be retrieved for participants to download and the test data set can be compared to the participants uploaded solution data to produce metrics. The pages in (4), (5), and (6) are system user interface (UI) elements that allow the users to upload or download TS data depending on the operational scenario.

One of the early decisions during our planning and design process was to decide to use the Django Python framework for implementing our system. Django uses a software design pattern similar to model-view-controller (MVC). However, instead of MVC, Django modifies this to be model-template-view (MTV): a view gets data from the models (which are stored in a database) and passes the data to templates, which in turn are displayed to the user. Because this pattern is at the core of the framework we are using to create this system, we thought it was important and worthwhile to mention it in the system overview. It is also worth noting that we will keep this pattern in mind when designing many of the other system components. If a "model" is mentioned, we define this as a class that holds data that is stored in the system's database. If a "view" is mentioned, we define this as a function that gets data from the models or from the user.

3 Software Architecture

3.1 Component Description and Functionality

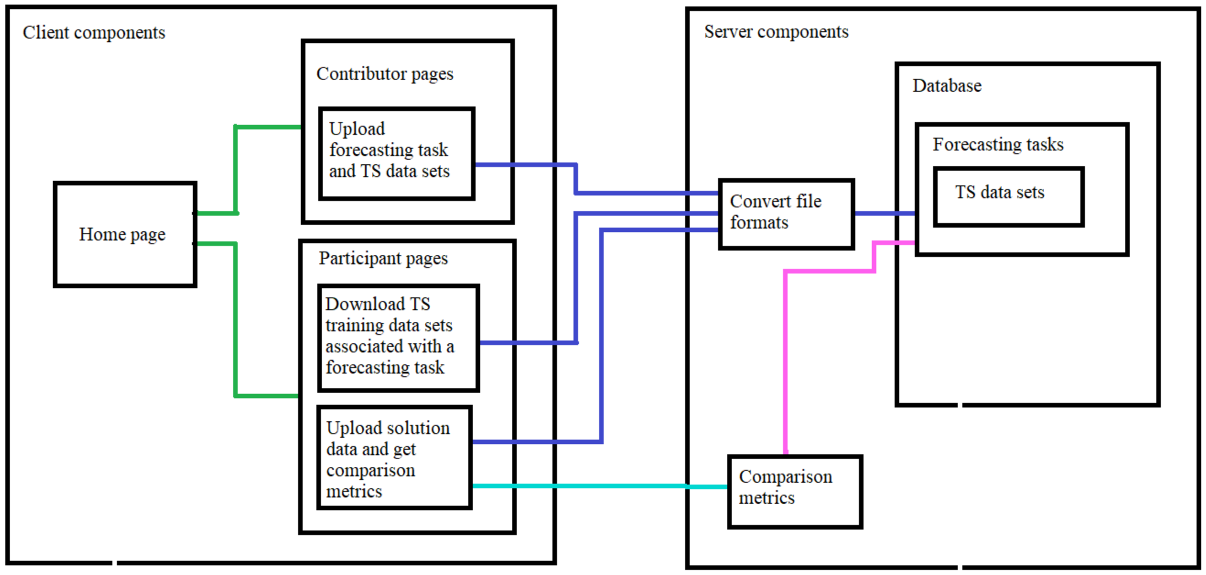
As listed in section 2, our system will have six main components. In this section, we will describe each component, as well as divide any components into smaller modules if needed.

Despite the SDS template document discouraging us from using the terms "client" and "server," we will use these terms to organize our components at a higher level. There is not a specific client component nor a specific server component: the components are simply organized with which part of the system they interact with the most, and all of these components require some sort of crossover or interaction with other components in the system. Additionally, all of the client components have some sort of server component to serve the client the correct web page. So, from a client–server relationship perspective, we can divide our components into the following set:

- ***Client components***: The set of components that a user can see and interact with.
    - ~~***Login/sign up***~~ ***Home page***: ~~This component allows users to create an account or log in with an existing account. This is the initial page that the user sees upon opening the application. After logging in or signing up, a~~ This allows a user to choose to be either a contributor or a participant, which will redirect the user to the appropriate subset of pages depending on their choice.
    - ***Contributor pages***: This subset of components are only visible to users that elect to be a contributor.
        - ***Upload forecasting task and TS training and test data sets***: A contributor that has a forecasting task and training data for that task can enter their forecasting task and upload the files associated with that task. The task is stored in the database, and the TS data sets are converted to a single format used by the database and stored in the database and associated with that task (this can easily be accomplished using a relational database).
    - ***Participant pages***: This subset of components are only visible to users that elect to be a participant.
        - ***Download TS training data set associated with a forecasting task***: A participant that wishes to complete a forecasting task must download the TS training data that is associated with that task. The data must be archived to a ZIP file ~~converted to the format specified by the user~~, and then provided for download.
        - ***Upload solution data set for a given forecasting task and get comparison metrics comparing the solution to that forecasting task's test data set***: The solution data must be converted to a single format used by the database, the test data must be retrieved from the database, and the solution data and test data must be compared, using the comparison metrics component. These metrics must be then displayed to the participant.
- ***Server components***: The set of components that a programmer can see and interact with.
    - ***Database***: The database contains a set of components that must be stored on a long-term time frame. The database must have a relational schema so that the TS data sets can be associated with a forecasting task. All TS data that is stored in the database must be in a single, uniform file format (This is only true for data that needs to be sent to the comparison metrics. Otherwise, we can store files in the format that they were uploaded, and provide copies of those files in a ZIP archive for participants to download).
        - ~~***User accounts***: Each user must provide a valid email, unique username, and password. The user's email and password are private. The user's username is public.~~

        - ***Forecasting tasks***: A forecasting task is defined by a forecasting period and a number of forecasts. It will also be useful to have a description associated with that task. A forecasting task is meaningless without having TS data sets associated with it.
            - ***Associated TS data sets***: TS data sets are a file uploaded by a contributor. A TS data set may contain multiple TS, and hence multiple files. Each file contains data for an independent time variable, and some other dependent variable. Each file is either training data or test data. Each file has associated TS metadata. Along with the TS data, the database must store whether the file contains training data or test data, the TS metadata, and which forecasting task the file is associated with.
    - ***Convert file formats module***: This module converts to and from file formats. Because we allow contributors to upload data in various different file formats (see section 3.1.2 in the SRS)~~, and we allow participants to download data in various different file formats (see sections 3.1.3 in the SRS)~~, and our comparison metrics module only takes data in one format, this means that we can store a single file type in our database, and convert to and from that file type when a user ~~either~~ uploads ~~or downloads~~ a file. This function is implemented by the convert file formats module.
    - ***Comparison metrics module***: This module compares the participant's uploaded solution data to the test data for a forecasting task of the participant's choosing. The module produces metrics, statistics, and graphs made during this comparison,

We can also create a diagram showing each of the components, as well as its children and parent components. This diagram also uses colored lines to indicate an interaction between components (these interactions are detailed in section 3.2). Each color represents a different subset of interactions.


Client components
Contributor pages
Upload forecasting task and TS data sets
Home page
Participant pages
Download TS training data sets associated with a forecasting task
Upload solution data and get comparison metrics
Server components
Database
Forecasting tasks
TS data sets
Convert file formats
Comparison metrics

3.2 Component Interaction

Within this TS repository application, there is no single module whose functionality is more important than the others. Each component is equally needed to ensure that the system functions. The interactions have been highlighted in the diagram above by using colored lines between components. Each subset of interactions is indicated by a different color.

- ~~***Login/sign up and user account interaction**: When a user creates an account (signs up), this account and the relevant information (email, username, password) must be stored in the database. When a user logs in, their username and password are authenticated by comparing the entered username and password to the account information stored in the database.*~~
- ~~***Login/sign up***~~ ***Home page and contributor page/participant page interactions***: Depending on whether the user chooses to be a contributor or a participant, they are redirected from the ~~login~~ home page ~~upon successfully logging in~~ to the corresponding contributor or participant page.
- ***Interactions involving the convert file format module***: The convert file format module is used both to convert data from the valid upload file formats to the uniform database file format, and from the uniform database file format the a set of file formats from which a participant can download data. This interaction involves both the contributor pages and the participant pages, because all of these pages involve uploading or downloading files, and all of the files that are uploaded and downloaded must be converted either to or from the uniform database file format.
- ***Comparison metrics and TS data sets interaction***: Once the participant has uploaded their solution data, this data goes through the convert file format module so that it can be converted to a uniform file format and sent to the TS data sets in the database, before being sent to the comparison metrics module. The comparison module also gets the test data from the TS data sets that is associated with the same forecasting task that the solution data is associated with. The comparison metrics module then produces the comparison metrics, statistics and graphs.
- ***Upload solution data and comparison metrics interaction***: The comparison metrics module produces the comparison metrics, statistics and graphs. These are sent to the upload solution data participant page and subsequently displayed.

3.3 Rationale for Architecture

The system's primary goal is to be an easy-to-use repository for time series data. The design decisions that reflect this can be seen through the number of components that are required, the hierarchy of components, and some specific reasons that can be seen through several components.

We tried to reduce the number of components to the minimum number for several reasons. The first was to make the system as simple as possible. This helps increase the maintainability, reliability, and portability of the system. When there are fewer components, this in turn leads to fewer interactions and reduces the complexity of the system. The relatively low number of components also means that it is easier to document each component's functions and its interactions with other components.

We create a hierarchy of components to organize each component and its interactions. This helps organize several things: the design decisions within each component, the implementations of the design decisions, and the documentation. This organization is also to separate modules whose functionality is independent from one another. This increases the maintainability and reliability of the system.

The system has several specific components that increase the usability of the system. The first of these is the convert file format module. This component means that the user can upload ~~and download~~ a file in any popular file format. Our system does not care about the types of files that are uploaded ~~or downloaded~~ because the convert file format module can convert them to ~~and from~~ a uniform file type that is stored in the database. The comparison metrics module is important for increasing the maintainability and reliability of the system. Instead of having a participant upload their ML model or forecasting algorithm, we instead have them upload their solution data. It is difficult to create a module that would accept an arbitrary model coded in an unknown language and produce metrics using that. Instead by having the participant upload their solution data, our system can avoid the implementation headache of having a function use the participant's model, and instead we simply compare their solution data to the test data uploaded by the contributor.

4 Software Modules

This section lists and describes the modules and components included in the system. Each module will be described by its primary role and function, its interfaces, a static and dynamic model, and design rationale.

4.1 Login/sign up

4.1.1 Primary Role and Function

~~The login or sign up page will allow users to create a new account or login with an existing account. If the user does not have a registered account, they will be asked to supply a valid username, email address, and password to gain access to the application. If the username is not taken, or the email is invalid, the user will be asked to try a different email or username to create their account. If the user already has an account, they will be able to login. If the user's login credentials are invalid, the user will receive a notification that their credentials do not match those of an existing user. After logging in,~~ a user selects if they want to be a contributor or a participant, and depending on that decision they are shown either the contributor page or the participant page.

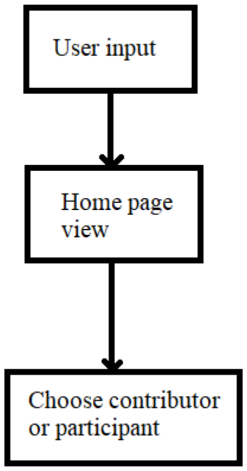
4.1.2 Interface

~~If the user is a past user or a new user, they will be directed to the login/sign up screen. The user has the option to either create an account or login with an existing account. If the user is new and needs to create an account, this interface will have a place for the user to input a username, email, and password. If the user is logging in, this interface will have a place for the user to input a username and a password.~~

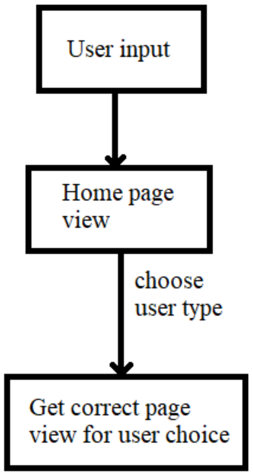
~~The login/sign up module also needs to interface with the database's user accounts. When a new user creates an account, their credentials are stored in the database. When a user logs in, the user's credentials are compared with the credentials in the database to check the inputted information's validity.~~

~~Once~~ the user ~~has successfully logged in, they~~ will need to input if they are logging in as a contributor or a participant. This input is passed to the server, and the corresponding pages are shown depending on the user's choice.

4.1.3 Static Model



4.1.4 Dynamic Model



4.1.5 Design Rationale

~~To make the login/sign up module simple, we decided to separate the "logging in" from the "signing up." This was an easy way to separate concerns, and could be easily implemented by creating two different paths for users to follow, depending on if they are a new user and need to create an account, or if they are a returning user and need to login. Once a user's account is created, they will be redirected to the login page to actually gain access to other areas of the system. This view was broken down into smaller steps to make the system easier to implement for the programs, and to increase the usability for the user.~~

4.1.6 Alternative Designs

We floated the idea of having a way to verify the user's email by actually sending them an email, but we decided that this was too ambitious and outside of the scope of the project.

Another design would do away with the login/sign up page altogether and would eliminate user accounts. The login/sign up page would be replaced with a page where the user could select if they are a contributor or participant, and the relevant pages would be displayed to the user based on their choice. This is the approach we eventually moved forward with and implemented.

4.2 Contributor task creation and TS data uploads

4.2.1 Primary Role and Function

The contributor task creation and TS data upload page will allow the contributor to add a forecasting task to the system and upload relevant data for that task. To do so, the user will enter the necessary inputs to create a forecasting task, as well as upload the relevant files. The files that are uploaded must be in the correct format of either CSV, XLSX, ~~XLS,~~ or JSON (as specified in SRS 3.2.3). The user also enters ~~metadata for the TS set, as well as~~ metadata for each file that stores TS data.

In the database, a task model is created to save the forecasting task. For each file that is uploaded and relevant to the forecasting task, a document model is created and the file is converted to the uniform file format and saved in the document model, along with a pointer to the original file. ~~The originally uploaded files are deleted, and the converted files saved in the document model are the only copy.~~ This document model holds the TS data file and all the relevant metadata. The document model will also hold a foreign key to the task model so that the TS data can be associated with a forecasting task.

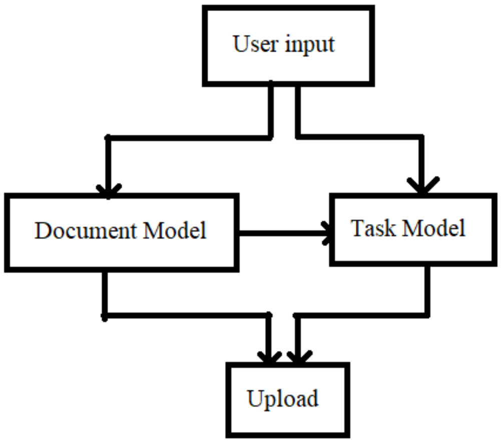
Using a relational database like SQLite (which is the Django default) allows us to create the relationships between the document model and the task model using a foreign key, and allows any number of files to be related to a forecasting task.

4.2.2 Interface

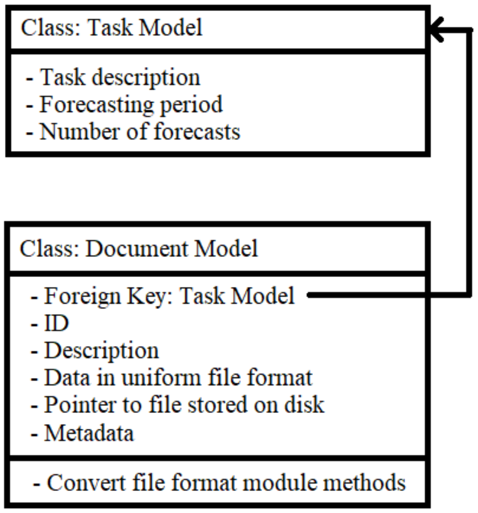
The interface for creating a task and uploading TS data will be fairly straightforward. The contributor will have to enter the required information for creating a forecasting task. The contributor will then have to upload all of the files that are relevant for that class, separated into two sets: the test data set and the training data set. ~~The contributor will then have to enter all of the relevant TS set metadata.~~ For each file, the user will then have to enter the TS metadata for each time series.

The forecasting task needs to be stored in the database first, so a task model is created to store the task. This is so that it can be referenced by each relevant TS data file. Each file will then need to interface with the convert file format module so that the file is converted into a uniform file format before it is stored in the database as a document model. For each document model that is stored in the database for the related task model, the document model needs to interface with the contributor upload page view that gets TS metadata from the user. This data needs to be entered into the document model and saved to the database.

4.2.3 Static Model

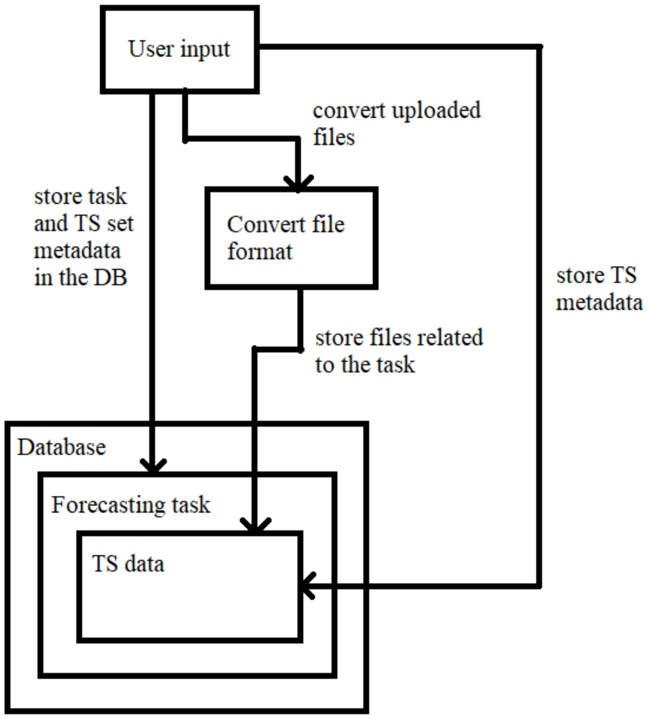


4.2.3.1 Database Relational Diagram



A document model can be related to only one task model, whereas a task can have many related documents, creating a one-to-many relationship (one task with many documents).

4.2.4 Dynamic Model



4.2.5 Design Rationale

To make the contributor upload module usable, we needed to have some sort of scheme where users can upload both univariate TS data sets and multivariate TS data sets for a single forecasting task. We assume that a TS data file contains only two columns: time and a dependent variable. So, with these assumptions, if a user wants to upload a multivariate TS data set, this means that they need to upload multiple files.

This is where a relational database scheme is useful: by having each document model be related to a task using a foreign key (see 4.2.3.1), we can have an arbitrary number of TS data files for a single forecasting task.

We also decided to allow users to upload any type of common data file (these types are mentioned in 4.2.1) to increase usability. However, to allow this ease-of-use feature, we needed a module that would convert from the uploaded file formats to the file format used in the database (we decided to use JSON as our uniform file format in the database). This convert file format module will be looked at in more depth in 4.7. The converted files are used by the comparison metrics module, but the original unconverted files that are saved are used to provide to the participants to download.

4.2.6 Alternative Designs

One other design that we thought about was having the contributor upload an additional metadata file along with their TS data. This file would require a certain format and would include the forecasting task and the metadata for every TS data file. We decided to not follow this route because we felt that it reduced the usability of the system, and possibly made the user a larger point of failure for our system: if the user-written metadata file was incorrectly formatted, this could create errors for our system when it is trying to use the uploaded data with different components.

4.3 Participant downloads TS training data

4.3.1 Primary Role and Function

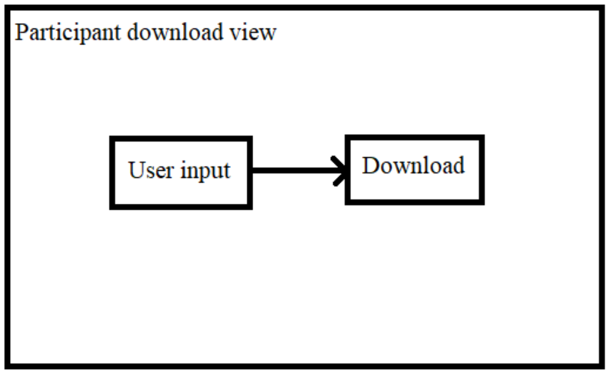
The participant TS training data download page will allow the participant to find a forecasting task that interests them based on the keywords in the task description. ~~The user can then select the file format that they wish to receive the downloaded files as.~~ The files that are related to the forecasting task that the user chose are then fetched from the database. The files are ~~converted to the desired format using the convert file format module, temporarily saved, and~~ archived in a ZIP file delivered to the user. Once the user has downloaded the files, the ZIP file is deleted.

4.3.2 Interface

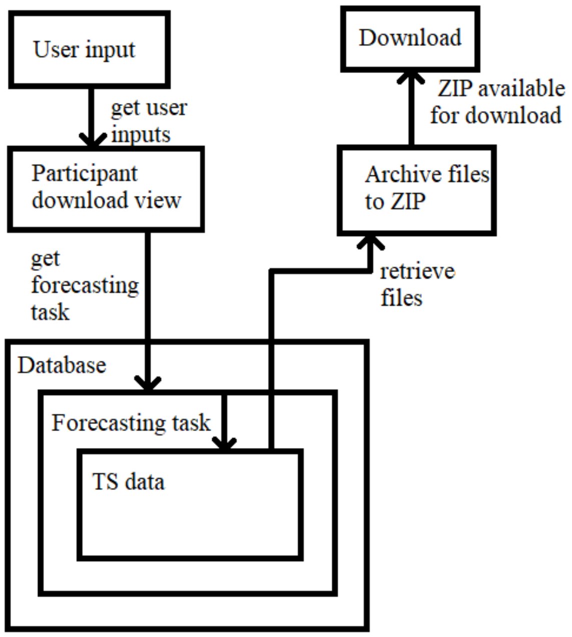
The interface for downloading TS training data will be fairly straight forward. The participant will have to select the forecasting task for which they want to download files. ~~The forecasting tasks will be organized by keywords found in the task description.~~ ~~The participant will then have to select the file format that they wish to receive the downloaded files as.~~ The participant will then click a button to download the files related to the forecasting task in the specified file format.

To do this, the participant TS training data download view will need to interface with the database to list all of the forecasting tasks and display them to the user. Once a user has selected a forecasting task ~~and specified a file format~~, the same view will need to get the files related to the task from the database and convert them using the convert file format module.

4.3.3 Static Model



4.3.4 Dynamic Model



4.3.5 Design Rationale

~~To increase the usability of the participant download module, we needed a scheme where users can choose their preferred data format to download. We can use the convert file format module to do this. This also is useful for the maintainability of our system, having one module that is useful in many applications decreases the complexity and allows future software engineers easier access to and understanding of the implementation.~~

~~The use of the relational database is also another design factor that makes it easier for the software engineer to implement this module. Rather than displaying every single file that is saved in the database—which would be confusing for a user who would have to look through every single TS training data file to find the ones that they are interested in—we instead display only the forecasting tasks, and can deliver the training data that is related to that task to the participant because of the relational database scheme.~~

We decided to create a ZIP archive of all of the files related to a task without converting any of the files to a user-specified format, and delivering that ZIP to the user to download. While this does decrease the usability for the end user, this makes the implementation much easier. ~~We decided against this because it would decrease the usability of the system and lower the ease-of-use on the user's end.~~

4.3.6 Alternative Designs

~~Before we decided to focus on the usability of the system as a major system attribute, another design for the participant download page involved creating a ZIP archive of all of the files related to a task without converting any of the files to a user-specified format, and delivering that ZIP to the user to download. We decided against this because it would decrease the usability of the system and lower the ease-of-use on the user's end.~~

To increase the usability of the participant download module, we needed a scheme where users can choose their preferred data format to download. We can use the convert file format module to do this. This also is useful for the maintainability of our system, having one module that is useful in many applications decreases the complexity and allows future software engineers easier access to and understanding of the implementation.

The use of the relational database is also another design factor that makes it easier for the software engineer to implement this module. Rather than displaying every single file that is saved in the database—which would be confusing for a user who would have to look through every single TS training data file to find the ones that they are interested in—we instead display only the forecasting tasks, and can deliver the training data that is related to that task to the participant because of the relational database scheme.

4.4 Participant uploads solution data

NOTE: This was the final integration of our step in our project, which we unfortunately were unable to deliver functionality for the system to deliver the metrics to the user.

4.4.1 Primary Role and Function

The participant TS solution data upload page will allow the participant to upload the solution data that they created by completing a forecasting task on the related training data. The participant will need to select the forecasting task for the data that they are uploading. The participant will then upload their solution data. This data gets converted to a uniform file format, and along with the test data that is retrieved from the database, sent to the comparison metrics module.

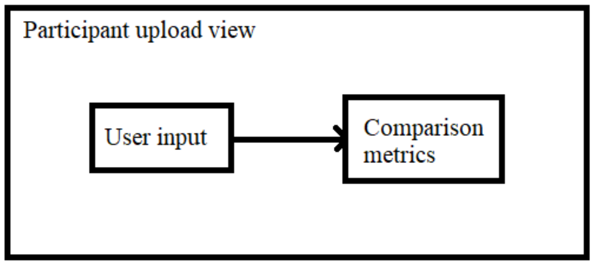
The comparison metrics module compares the solution data and the test data, and creates error metrics, statistics, and graphs comparing the data, which would be downloaded in a ZIP archive by the user. The comparison metrics module will be explored in more depth in 4.8.

4.4.2 Interface

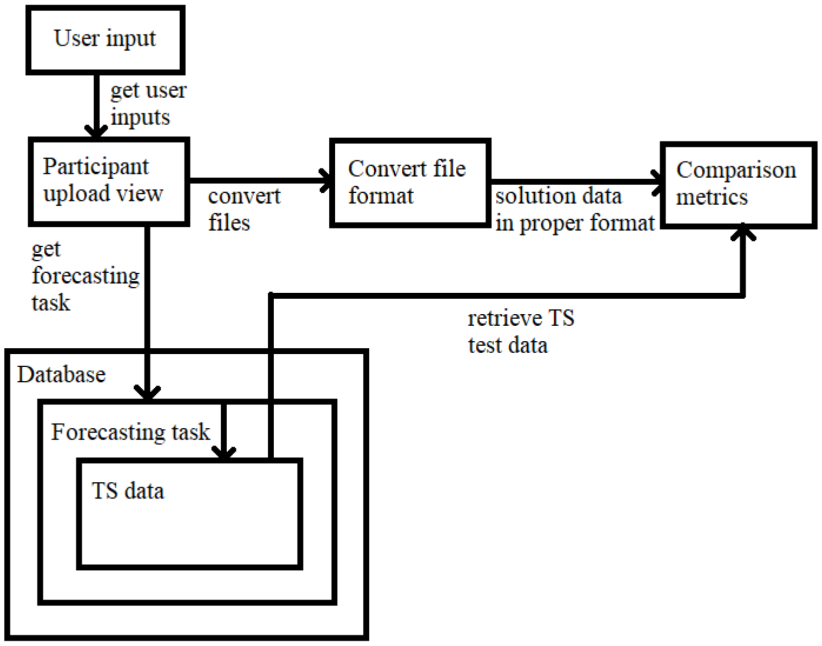
The participant TS solution data upload page will need to interface with the database and several other components. The participant first needs to select the forecasting task for which they are uploading solution data. The participant will then upload their solution data. A participant upload view will have another interface where it fetches the forecasting task from the database and queries the related TS test data sets, sending them to the comparison metrics module.

The participant upload view will also need to interface with the convert file format module, where it will convert the uploaded solution data into the uniform file format. Once this is complete, the participant upload view will interface with the comparison metrics module, sending the uploaded and converted solution data to the comparison metrics module.

4.4.3 Static Model



4.4.4 Dynamic Model



4.4.5 Design Rationale

We again use the convert file format module to allow users to upload their solution data in any file format, increasing the usability of our system. The solution data gets sent to the comparison metrics module after it is converted to the uniform file format. The participant also selects the forecasting task which retrieves the TS test data from the database, making use of the relational scheme.

Our primary design goals were to make this module easy for the user to use, as well as making use of the modules and relations that were already designed.

4.4.6 Alternative Designs

One design consideration that we made when designing our system was to have the comparison metrics be written to a file that the participant could download. We rejected this design in favor of a design where the metrics were displayed on a web page, mainly due to the fact that displaying the metrics would make the system more immediately usable. In an ideal world, our system would both display the metrics and make them available for download, but we decided that doing both of these would be out of the scope of the project in the time that we had.

~~4.5 User accounts~~

~~4.5.1 Primary Role and Function~~

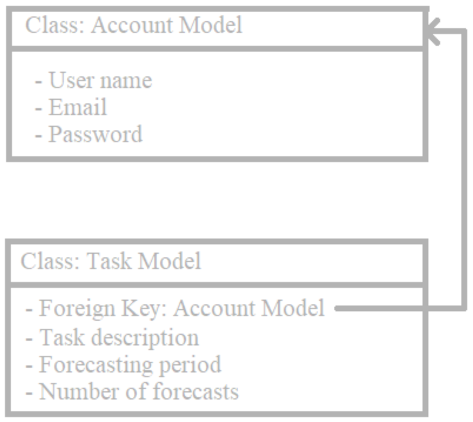
~~The primary role of the user account is to authenticate users and ensure that not any random person can upload data to the system. The user accounts also allow the system administrators to see who has uploaded what tasks and what data.~~

~~4.5.2 Interface~~

~~The user accounts need to interface with the login/sign up view as well as the database. The login/sign up view will get the inputted user account information. If the user is a new user, their account is created and stored in the database. If they are a returning user, their login information is compared with the account information that is stored in the database.~~

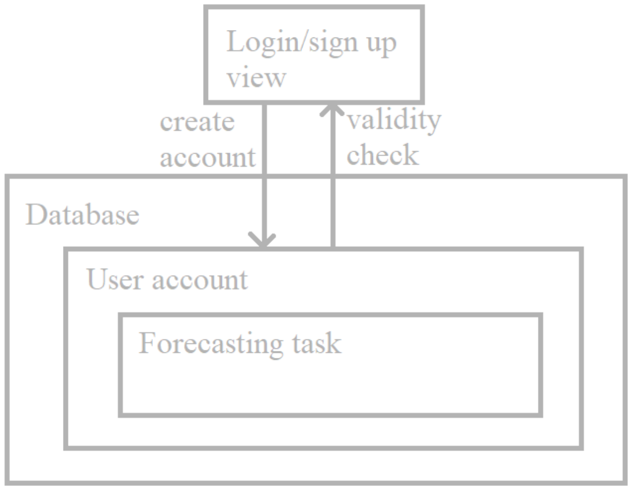
~~4.5.3 Static Model~~

This model was not used.



~~4.5.4 Dynamic Model~~

This model was not used.



~~4.5.5 Design Rationale~~

~~Our system uses user accounts so that there is a small barrier to entry to use the system. If there was no account system, it would be very easy for a malicious user to upload unusable data to the system. With an account system, this will discourage the lazy malicious users from uploading unusable data.~~

4.5.6 Alternative Designs

Alternatively, we could not include accounts at all, and make our system available to any user regardless of account. Instead of having a login/sign up page, there would be a home page where the user would select if they are a contributor or participant, and then be redirected to the appropriate pages depending on their choice. We decided to not use this alternative because of the points laid out in section 4.5.5.

We ended up using this alternative design, as it became clear during our implementation that adding user accounts would be extra work, as well as being outside of the main scope of functionality of the system.

4.6 Forecasting tasks

4.6.1 Primary Role and Function

The primary role of the forecasting task is to give the participants the outline for a task to complete, and what solution data to produce. It also will function as a way for users to find a task or a data set that interests them by having associated keywords in the forecasting task description.

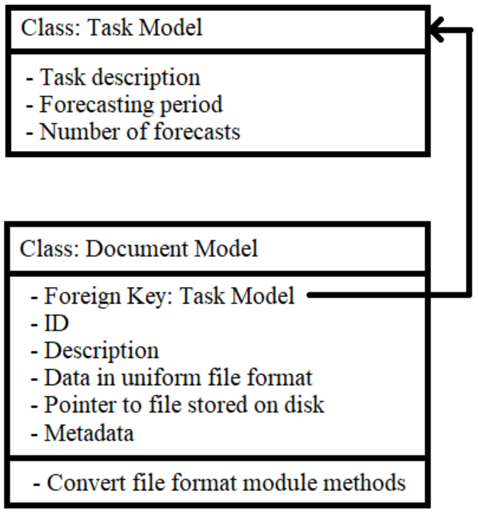
The forecasting task will also function as a foreign key for the related TS data that a contributor will upload for that task. This means that a user will only have to look through the forecasting tasks to find a task/data that interests them, instead of having to look through every TS data file that has been uploaded to the database.

4.6.2 Interface

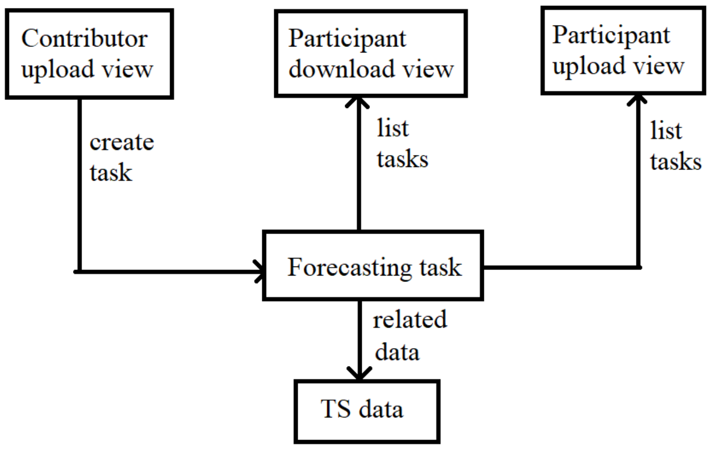
The forecasting task has to interface with several client components, to accept and create a new forecasting task provided by a contributor, or to provide a list of all the saved forecasting tasks to the participant that wants to either download the TS training data or to upload their solution data for a specific task.

The forecasting task also has to interface with the database~~, the user account model,~~ and the document model. The forecasting task will store a foreign key to ~~the user account model that created that task, and~~ each TS data file that is stored in a document model and is related to that forecasting task~~ will have a foreign key to that task mode~~l.

4.6.3 Static Model



4.6.4 Dynamic Model



4.6.5 Design Rationale

We used a relational database to allow our design of the forecasting task model to have relations between the forecasting task and the ~~user account model and the~~ TS data document model. This allows us to have a forecasting task related to an arbitrary number of files. ~~Additionally, this allows a forecasting task to be related to a user.~~ Lastly, this makes it easy for the programmer to list all of the forecasting tasks, and allows the user to download the TS data associated with a forecasting task, instead of having to browse through every TS data file that is stored in the database.

4.6.6 Alternative designs

We did not come up with any alternative designs for storing the forecasting task and the TS data in our database. Our initial design using a relational database and foreign keys worked, and was simple and easy to implement using Django's models and its backend database system.

4.7 Convert file formats

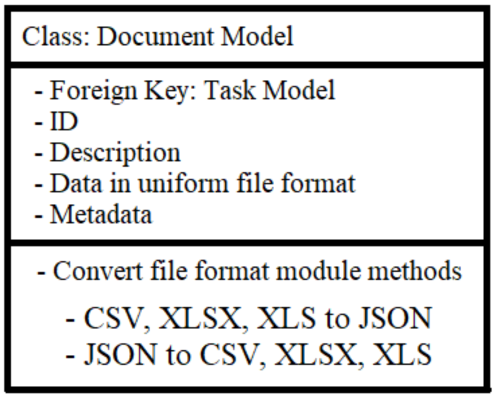
4.7.1 Primary Role and Function

The primary role of this module is to convert file formats to and from the uniform file format that is used to store TS data in the database. More specifically, we allow users to upload ~~and download~~ files in multiple formats: CSV, XLSX, ~~XLS~~, and JSON. However, we only want to store the files in a uniform type in our database, this format will be JSON. To do this, the convert file format module will implement functionality to convert to CSV, XLSX, and ~~XLS~~ from JSON and vice versa. These convert file format functions will be implemented as methods of the document model, and the appropriate methods will be called to convert TS data to and from the uniform file format.

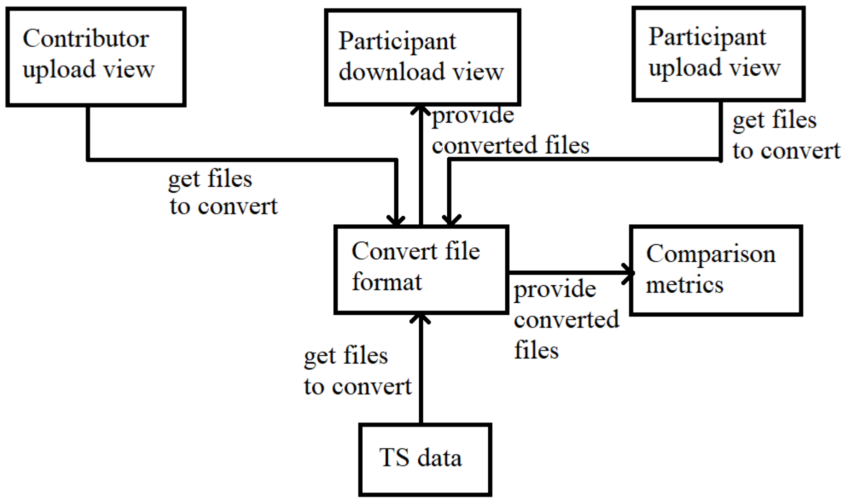
4.7.2 Interface

The convert file format module will need to interface with the TS data to ~~convert to and~~ from that data. It will also need to interface with the contributor upload view and the participant upload and download views so that users can upload ~~or download~~ TS data sets in whatever format they desire.

4.7.3 Static Model



4.7.4 Dynamic Model



4.7.5 Design Rationale

We designed the convert file format module to increase the usability of the system. This allows users to upload ~~and download~~ files in the formats of their choosing. This convert file format module interacts with many of the other components of the system. Because of the system's usability requirements, this module is needed to make the user's experience easier, and we thought that meeting this requirement would be worth any extra implementation difficulties.

4.7.6 Alternative Designs

Another possible architecture would have been to exclude the convert file format module, and simply do not convert any of the files. This would have made comparison metrics more complicated, at least in terms of how the data is delivered to the comparison metrics module, but it would have made uploading and downloading the files simpler, as the system would not have to convert file formats when uploading or downloading files. We decided not to go with this alternative design to increase the system's usability attributes.

4.8 Comparison metrics

NOTE: This was the final integration of our step in our project, which we unfortunately were unable to deliver functionality for the system to deliver the metrics to the user. Our current working version instead uses randomly generated "fake" solution data to compare to the contributor's uploaded test data.

4.8.1 Primary Role and Function

The primary function of the comparison metrics module is to compare the solution data uploaded by a participant to some TS test data set uploaded by a contributor. These two sets of data would be both related to the same forecasting task. The comparison metrics module then calculates the two sets of data, creating statistics and graphs that are useful to the participant. These include:

- Accuracy score, the percent correct,
- Pearson correlation coefficient, r, the relationship strength,
- Mean absolute error (MAE), the average difference,
- Mean absolute percent error (MAPE), the average percent difference,
- Symmetric MAPE (SMAPE), the average percent difference from a symmetric mean,
- Mean squared error (MSE), the average difference squared, and
- Root MSE (RMSE), the square root of the average difference squared.
- Data comparison graphs, showing the training data, test data, and solution data.

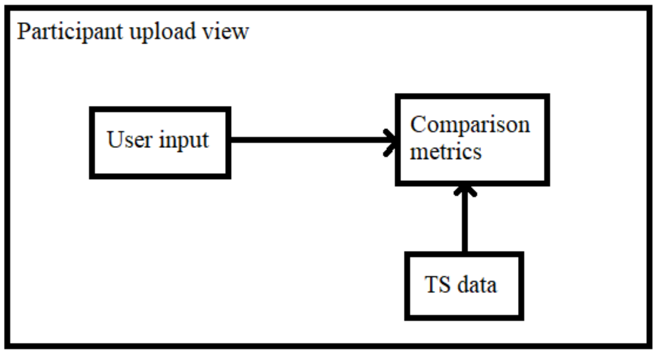
These metrics will all be ~~displayed to the user.~~ made available for the user to download in a ZIP archive.

4.8.2 Interface

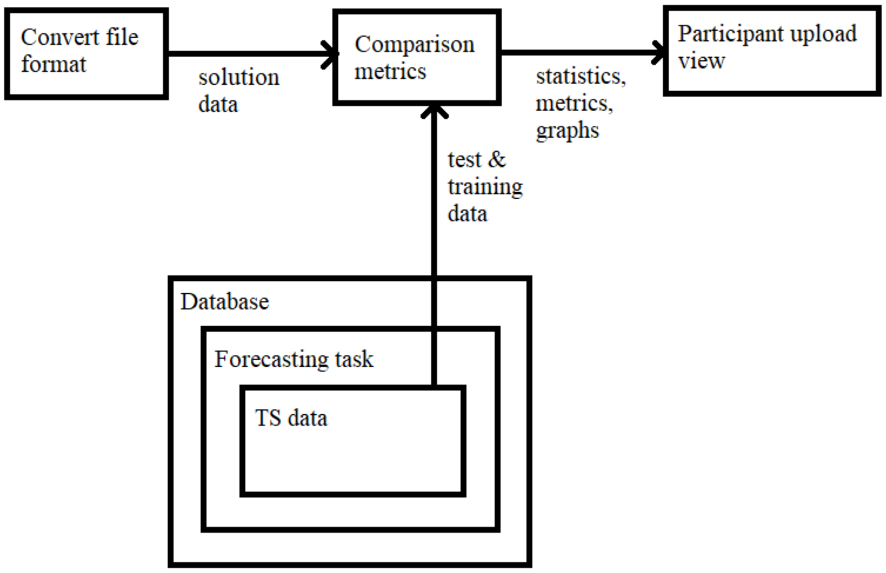
The comparison metrics module receives three inputs: the solution data, the test data, and the training data. These are retrieved from the database in the case of the test and training data, or from the convert file format module in the case of the solution data.

After producing the metrics, statistics, and graphs, the comparison metrics module interfaces with the participant upload view so that the metrics and graphs that it has produced can be ~~displayed to the user~~ downloaded by the user in a ZIP archive.

4.8.3 Static Model



4.8.4 Dynamic Model



4.8.5 Design Rationale

The comparison metrics were part of the project description, and the actual implementation of getting the statistics will be straightforward using a Python library like Numpy or Pandas. The real design decisions involved figuring out what data the comparison metrics would accept, as well as how those metrics would be presented to the participant. The comparison metrics module takes in the solution data and both the test and training data so that a comprehensive graph can be created showing the data that the forecast was trained upon, the data that the forecast produced, and the actual data to compare with the forecasted data.

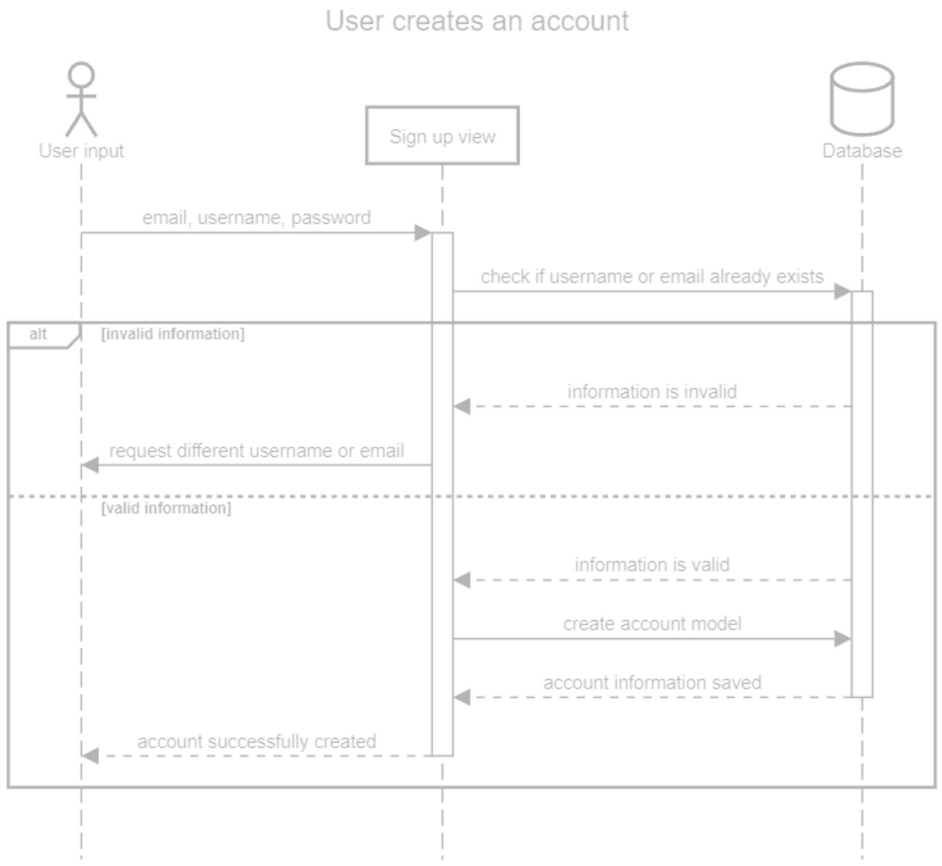
4.8.6 Alternative Designs

There were not many alternative designs that we considered for the comparison metrics module, mainly due to the fact that this module's functions were straightforward in terms of software requirements.

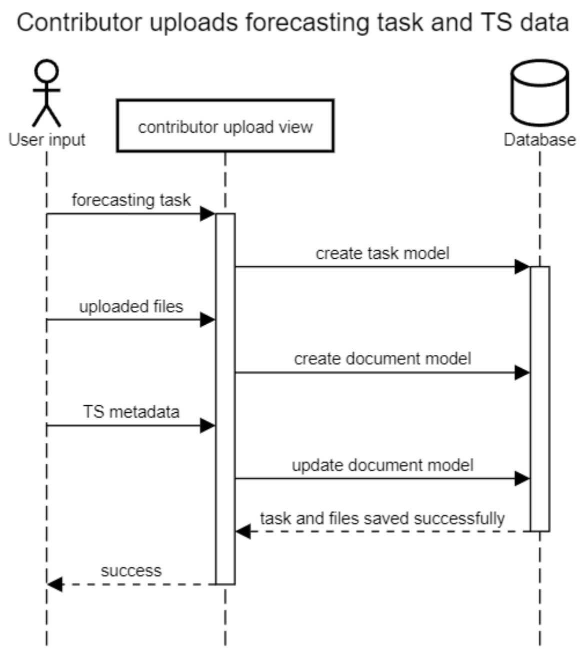
5 Dynamic Models of Operational Scenarios

~~5.1 User creates an account~~

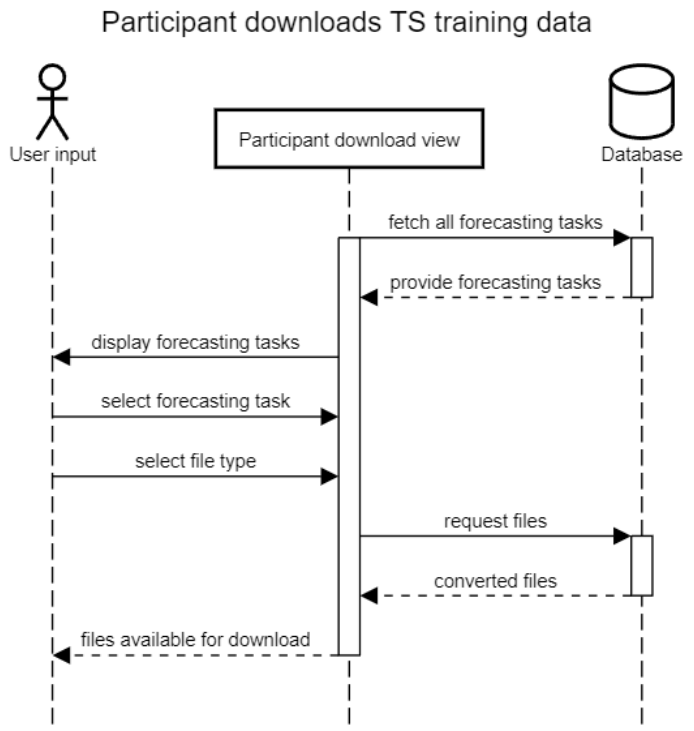
This model was not used.



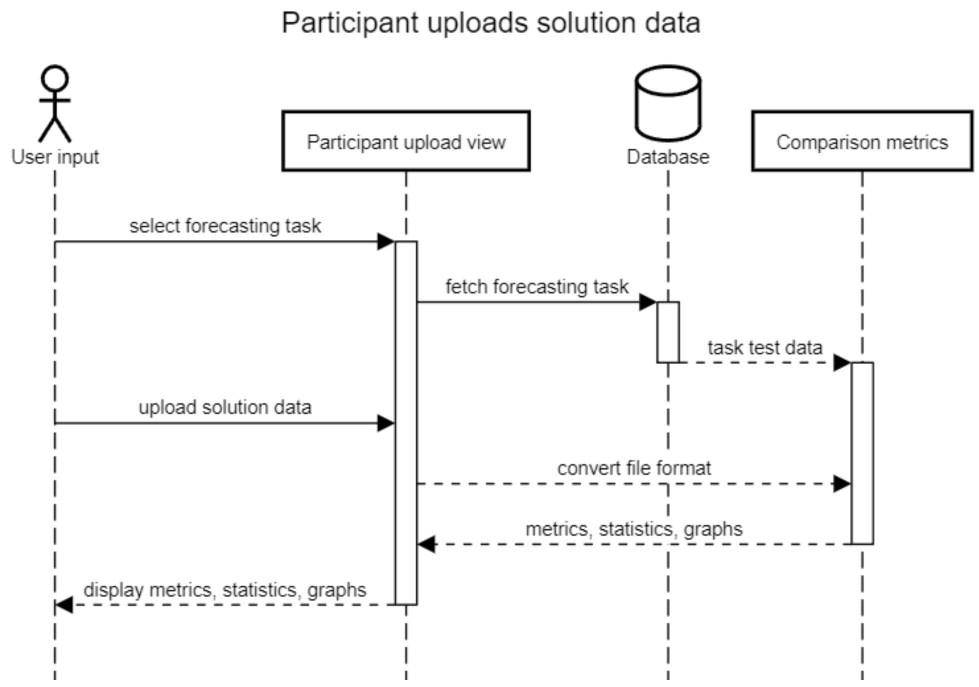
5.2 Contributor uploads forecasting task and TS data



5.3 Participant downloads TS training data


Participant downloads TS training data
User input
Participant download view
Database
fetch all forecasting tasks
provide forecasting tasks
display forecasting tasks
select forecasting task
select file type
request files
converted files
files available for download

5.4 Participant uploads solution data



6 References

Flores, Juan. (2023). CS 422—Software Methodologies 1—Project 1: Time Series Forecasting and Benchmarks.

Sethi, Ravi. (2023). *Software Engineering: Basic Principles and Best Practices*, 1st ed., Cambridge University Press.

van Vliet, Hans. (2008). *Software Engineering: Principles and Practice*, 3rd ed., John Wiley & Sons.

Model–view–controller. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller.

Sequence diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Sequence_diagram.

Class Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Class_diagram.

SequenceDiagram.org. https://sequencediagram.org/.

7 Acknowledgements

The SDS template was provided by Dr. Juan Flores, who changed the original template document created by Dr. Anthony Hornof.

The provided example submission of the SDS document, authored by Ronny Fuentes, Kyra Novitzky, Jack Sanders, Stephanie Schofield, Callista West (https://github.com/JackSanders1998/CIS422Proj2).