COMP9900 Project Report

Submitted

by

Team FAB5

Team Members

Rong Zhang

Vandhana Visakamurthy

Yiligong

Yu Han

Yuehui Chu

Overview

Introduction

People who are passionate about reading and are books lovers are constantly looking for the next best book to read. A web based book recommendation site can help readers keep track of all books that they've read, books that they want to read, and get personalised recommendations to find the next book to read. It also aims to help users find like minded readers and explore their collections to browse some books they could possibly like. Though this is the primary aim of ReadRecommend, it is not limited to this. Users can create collections, search and filter searches, set a reading goal and keep track of their reading goal, search other users, view other users' collections and browse random books. Users can also rate and review books and add books that are beyond the database of ReadRecommend. Our team aims to provide all these services in an efficient and user friendly way to the users.

Project Description

The website must be a platform for readers to connect with like-minded readers and find new books to read. It should also allow readers to share their experiences and review books. Therefore, the reader must be able to login as a user to avail such services. The user must use his/her email and set a password to use the website. The user provides other information like age and gender to improve the recommendation experience. The user can search for books and view the summary and reviews of every book in the database. Users can create collections and customise them as named collections. Users can add books to these customisable collections and access them whenever. The user can also view other people's recently added/read books. Users must be able to see the total number of readers a certain book has. Users can review and rate books out of five and the average of the ratings are recorded and displayed when the book is searched for. Based on the user's selections, books must be recommended to the user of similar interest and with various recommendation modes. Apart from all of which, the user must be able to set a reading goal and a time period to achieve said goal.

System architecture

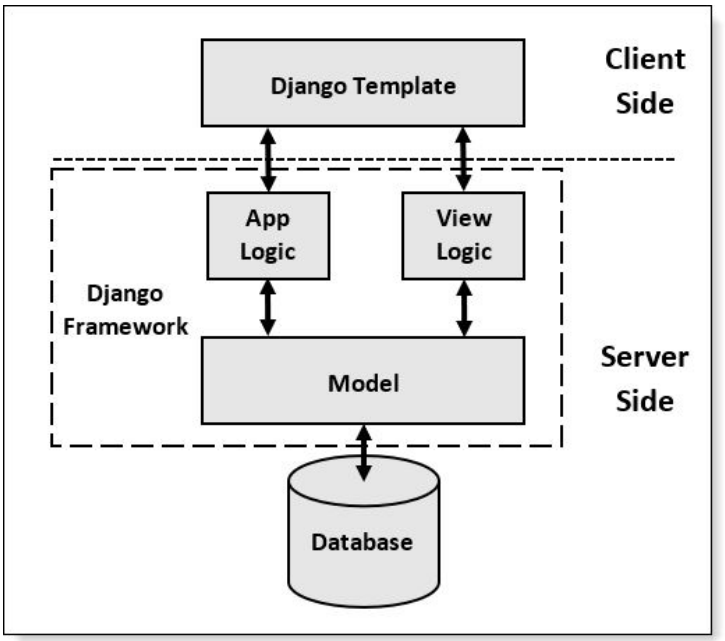


Figure 1. A holistic view of Django's architecture (George 2018)

This project is heavily laid on Django architecture. All displaying web pages come from Django Template. Transfering the data from the database has been laid on the server side which we decided to host on a Google VM instance. The database is also hosted on a Google MySQL instance.

Presentation layer

Users would use a web browser to access the service. Users can see the result on the web pages and interact by entering values in input fields and clicking buttons in order to use functions. This layer is designed and developed using HTML/CSS, Javascript, Bootstrap and Ajax when required.

Business layer

Django + Python Script: Django is a powerful frontend web tool with python. It can provide highly interactive web experience and comprehensive python libraries. By using python script, it can separate the front end and backend. It can handle the model and achieve data extraction, data storage, and recommendation systems. It connects presentation layer and data layer. Models and views are created using the Django template.

Data layer

In this project, we use google cloud SQL in order to store, analyse and extract books, collections, reviews, and user's data. It is connected by the business layer. The VM(Compute Engine Instance) is connected to the Google SQL Database. In localhost this would be SQLite.

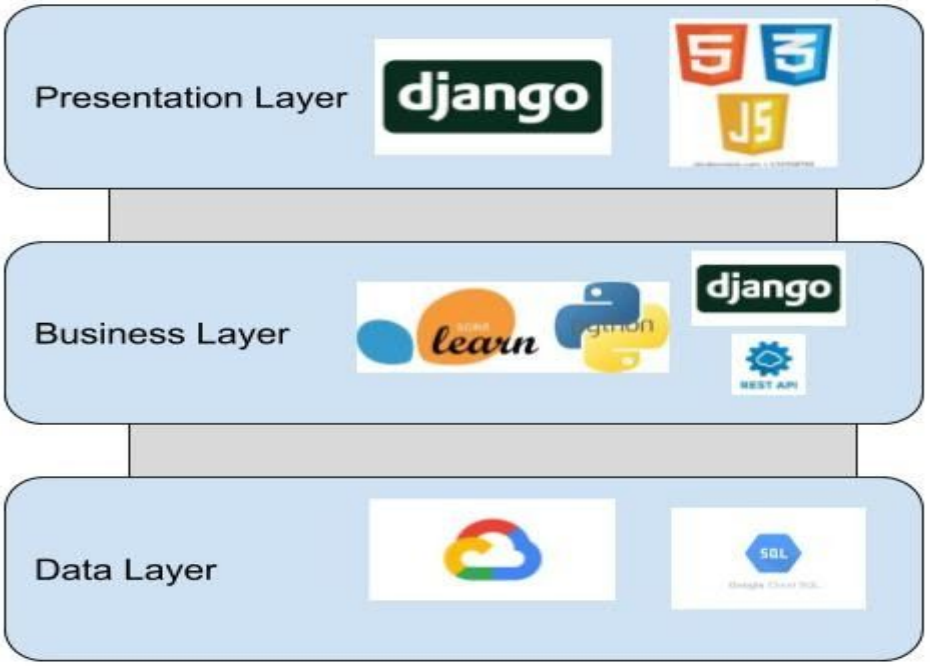


Figure 2. Layered view of the System Architecture

Functionalities and Implementation Challenges

Sprint 1

READ-1: As a user, I want to register an account so that I can use the services on this website.

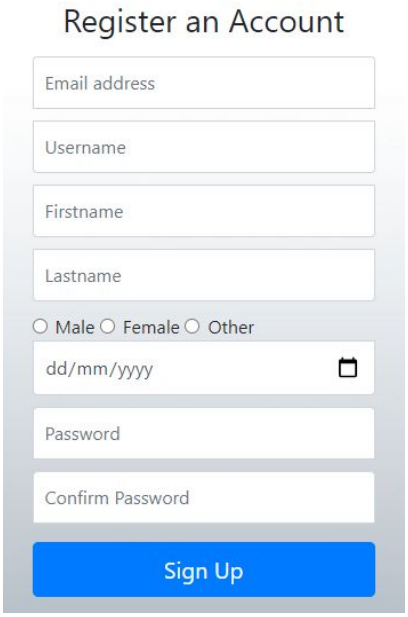


Figure 3. Register Page

In this user story, new users can register on our website. We used Django's authentication system to record users' info which include email, username, first name, last name, gender, birthday and password, and these information also can help the system to predict user behaviors in the later stage.

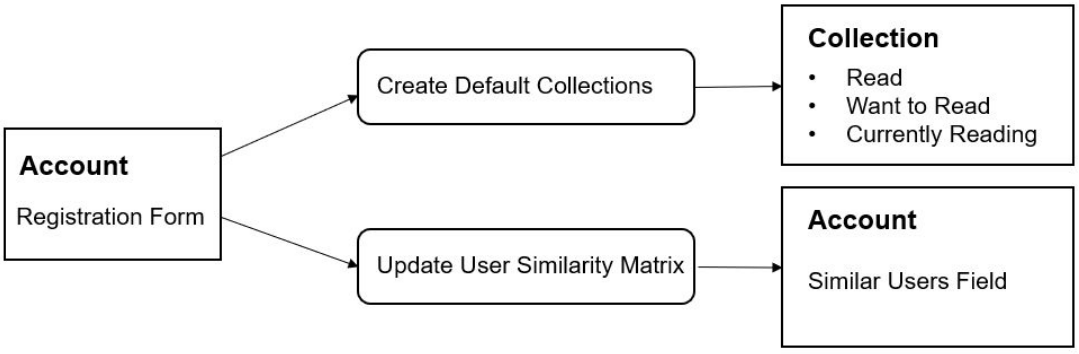


Figure 4. Registration View Function Workflow

There is a view function (registration_view) for this user story. Firstly, this function will check the validation of the registration request and save the new user info to the database. Then this function will create three default collections for the user which are “Read”, “Want to Read”, and “Currently Reading”. Finally, this function will use the new user’s info to calculate the user similarity and update similar users for all existing users. After finishing all tasks above, this function uses the email address and password to login automatically.

There was a problem when we added the similar users field under account model. We defined this field as list because we hope to use a list to save the similar user’s username, but we cannot save the target into our database. After reading the model field reference in the django document (django, 2020), we knew that we have to use correct field types under the model function, but list is not included in field types. Therefore, we change to use textfield to record the username and use ‘/’ as the delimiter, then we can save the similar users to the database successfully.

READ-2: As a user, I want to login into my account so that I can use the services on this website.

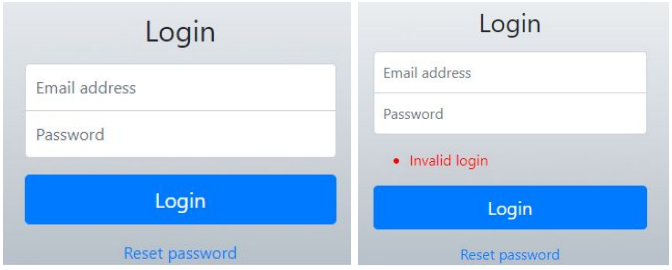


Figure 5. Login Page

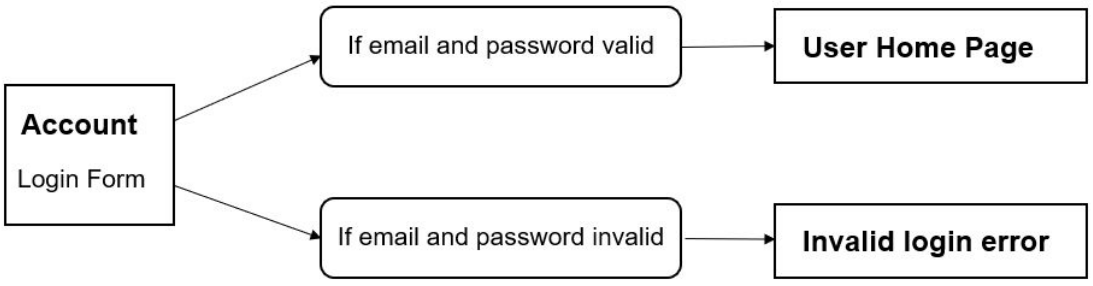


Figure 6. Login View Function Workflow

This function allows existing users to login the website and use all services. We used the password authentication function provided by Django to check

the email and password match or not. If the email and password are correct, the website will redirect to the user home page, otherwise the user will see the invalid login error on the login page.

Before doing this part, we didn't know how to print the error on the web page. We searched the django document about the form errors, then we learned that we can add the form.errors into the html page, and then the user can see the error if they are doing something wrong. Also we learned about how to raise validation errors which is supported by django.

READ-4: As a reader, I want to create a book collection so that I can classify books.

Figure 7 . The template of collections

This user story provides the functionality of creating a book collection for the reader so the reader can use the book collection to classify its books. It maps the project objective, which readers must be able to keep track of books they have read in one of many easily identifiable book collections on their account. The reader can add a collection with provided name and description on the collections page. Then, the reader can use it to store books which will be described in other user stories.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Model: There is a model of collection, which contains UID (foreign key to account), collectionName, description, createdDate, and books (foreign keys to book).
- Template: There is a view of a collection list, which includes a creating collection button and a list of collections. The list of collections will display the data from the controller.

- View: There is a view function called collections_view, which will accept the request and return the data of collections' name and description. It also saves the new collection into the database with the definition of the collection model.

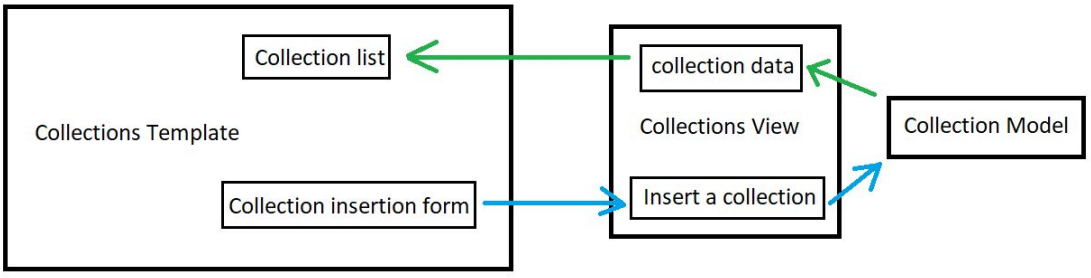


Figure 8 . Workflow of the collections user story

The challenge of this user story is about the inserting data into the database, which are used in many future user stories. Initially, we used the form HTML elements with POST. However, when it integrates with other functions, the invalid form submitted by the reader would be re-submitted. And directly displaying the form would take too much space on the webpage. In order to solve these issues, I decided to use jQuery responsive modal and AJAX to submit the POST request to view. There is a button which can call a modal window to display the insertion form of a collection. The view function will take care of the collection insertion and refresh the page in order to show the newest collection list.

READ-5: As a reader, I want to remove a book from my collections so that I don't need to track that book I don't like any more.

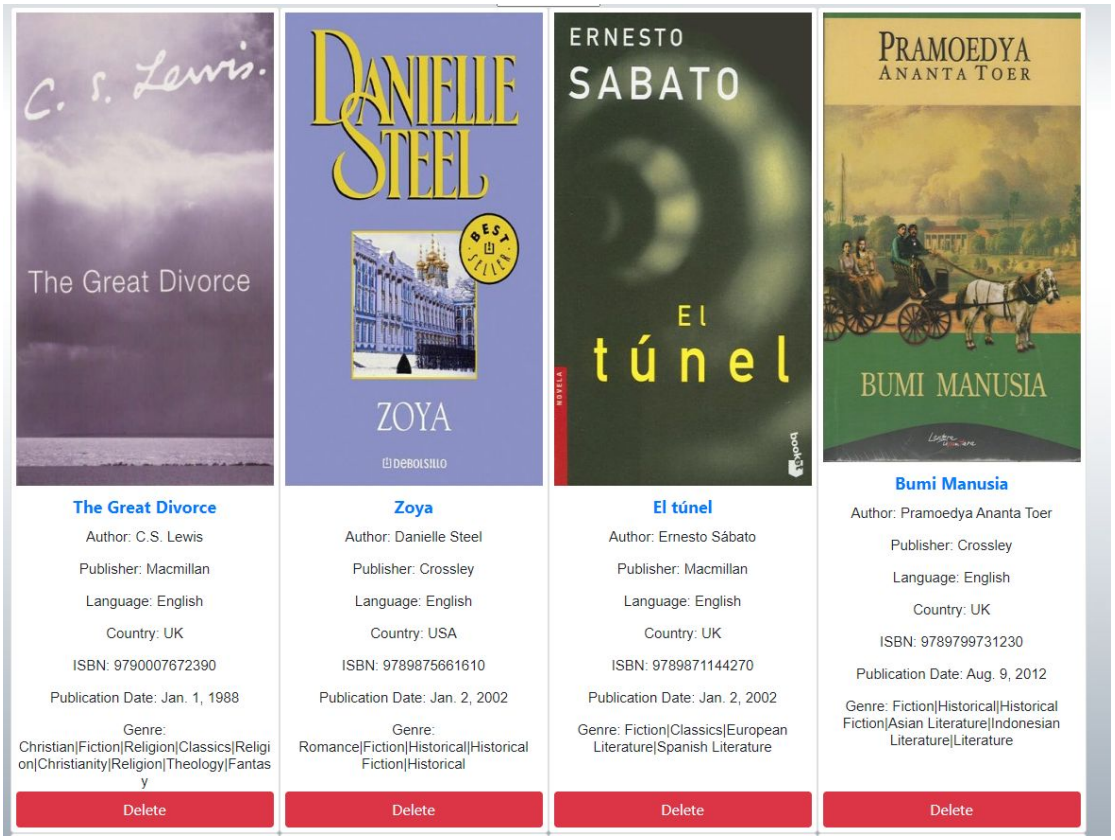


Figure 9 . The delete button on the collection page.

This user story provides the functionality of removing a book from a collection of the reader. This maps to the project objective that the readers must be able to add or remove books to/from any of their own collections. The reader can remove any books from the collection at any time by clicking the delete button.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Template: a simple link contains collection id and book id
- Model: the collection model stated in READ-4. In this user story, we are using the manytomany field books.
- View: There is a view function called book_delete, which will accept the request, collection id and book id. It will find the book reference with the book id and delete it from the collection's books field. After that, it will jump back to the collection page.

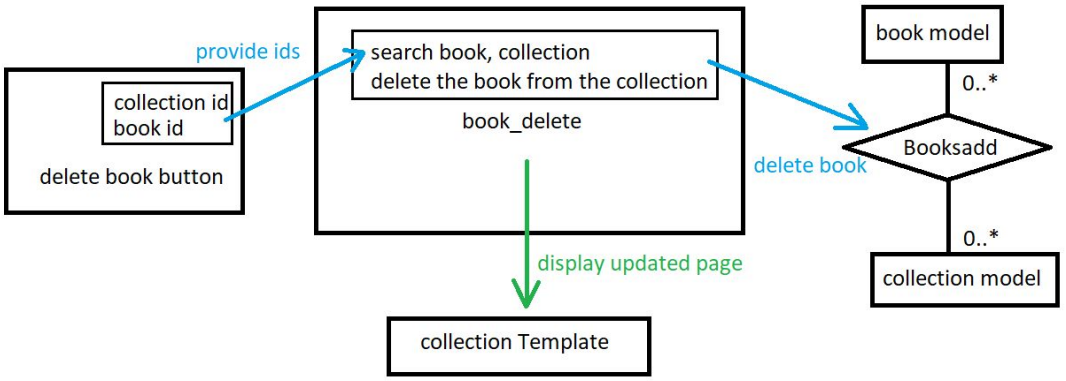


Figure 10. workflow of deleting a book from a collection by a reader

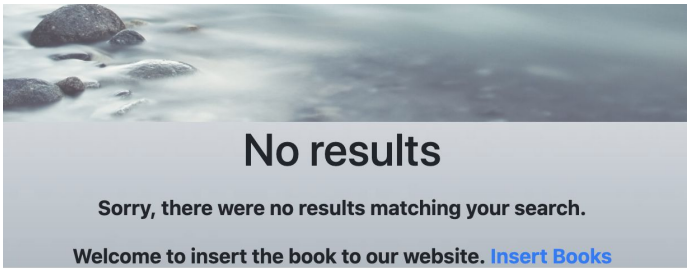
At this stage, we were not familiar with dynamic web content, therefore, how to ask the database to delete the book-collection record was a problem. The problem can be separated into two parts, first, how to transmit the information of the book and the collection, and second, how to update the view. I have created a link which contains the id of the collection and book. When clicking the link, it will pass it into the book_delete view, which will check and delete that book from the collection. However, during the implementation, there was another bug we found, that is any user can view the delete button of other users' collection page. In order to solve this problem, I added two protecting processes. First, remove the content from the template when the visitor is not the owner of the collection. Second, check whether the user id from the request is the same as the user id of the collection. It ensures only the owner can delete the books from their collections.

READ-36: As a reader, I want to add a book if it is not on the website so that I can record books I have read.

This functionality allows users to insert books that are not in the database. Initially, the team designed and developed the insertions using csv files. As per suggestions from the first progressive demo, the team redid this user story based on the feedback from our supervisors. The insertion is user friendly and allows users to add all details of the books and upload a book cover to insert a book.

This user story was difficult to implement because the images uploaded by the user weren't being displayed. This was because the MEDIA_ROOT and the path to store the images were different from the path from which the book covers were being retrieved. This was fixed after creating a separate folder for media access and offering two paths depending on the source of the dataset.

This functionality was implemented using Django custom form model, and is designed using bootstrap widgets and HTML/CSS styling. The book_insert view saves the information from the form in the database.



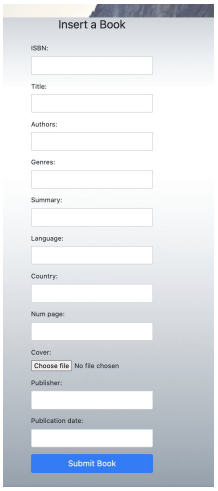


Figure 11. Search displays no results redirects user to Book Insert form

READ-67: As a user, I want to reset my password so that if I forget my password, I can get a reset email and set a new password.

The functionality which READ-67 provides is to reset a user's password when the user does not log in. First, the user should enter his registered email so as to send reset email to his mailbox then the email containing resetting password link will be sent and finally the user can click the link to set a new password.

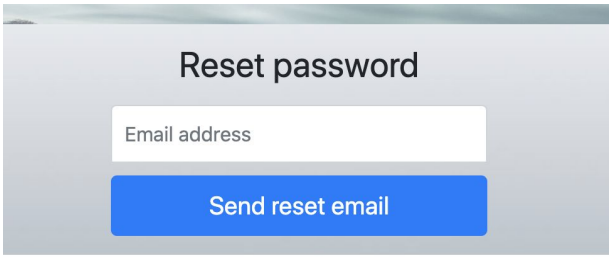


Figure 12 . Enter email to reset password

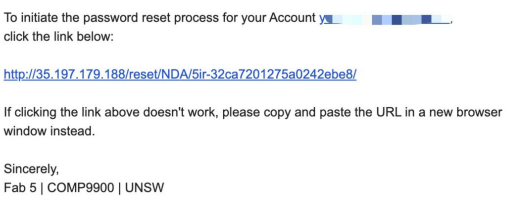


Figure 13 . The email for resetting password

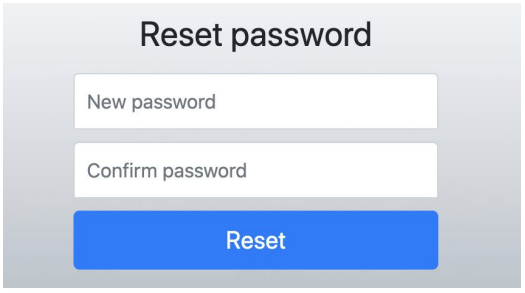


Figure 14 . The page of enter new password

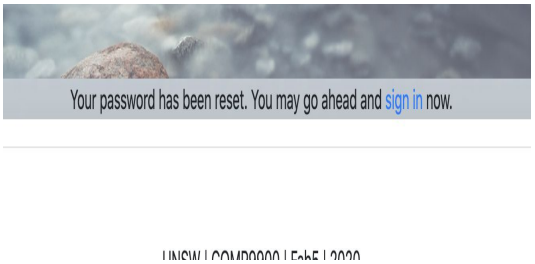


Figure 15. The page of resetting password successfully

The password reset functionality uses the views in django.contrib.auth, and they are:

- PasswordResetConfirmView
- PasswordResetView
- PasswordResetCompleteView

the templates used are:

- password_reset.html
- password_reset_complete.html
- password_reset_done.html
- password_reset_email.html
- password_reset_form.html

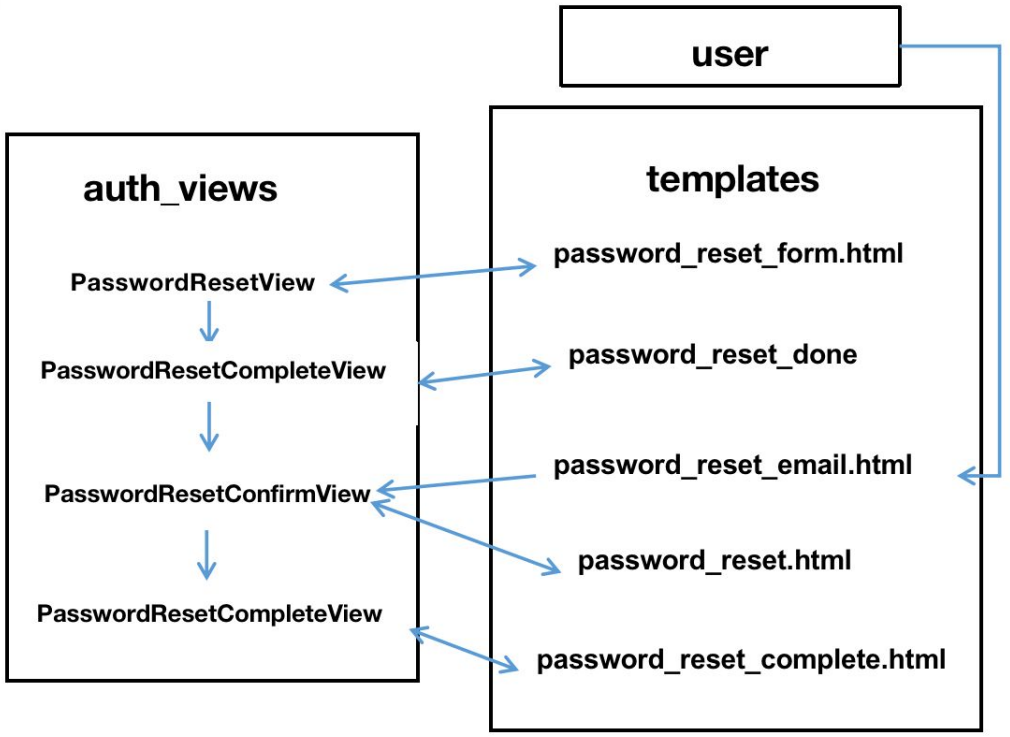


Figure 16. Password Authentication

Firstly, PasswordResetView will receive the reset password request and check the validity of the mail address from the form in password_reset_form.html. If the email address is valid, then password resetting mail is sent. Then PasswordResetCompleteView will call template password_reset_done.html, then some information like checking password resetting email in your email box will be shown. When the user clicks the password resetting link in email, PasswordResetConfirmView will receive the request and the new password in the form of password_reset.html will update the old password. At last PasswordResetCompleteView will call password_reset_complete.html to show the process of resetting password completely.

For development purposes Django lets us store emails in the console, so the challenge for this user story is how to configure our own SMTP server. After a period of time of learning, we choose mailgun for our mail server, and so next we learn how to create a mailgun account and configure it. The next step we learned is how to deploy settings.py in Django by our mailgun account.

Sprint 2

READ-7: As a reader, I want to look through all possible books I like so that I can find my favorite as soon as possible.

The functionality which READ-7 provides is to show all possible books in the database, then the user can choose one he likes as soon as possible. READ-7 maps to the object objectives “Readers must be able to view the full details of any book, (including all reviews), in any book collection”.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Template:the card class in books_list.html is used to show book cover, title, language and so on.
- Model: the model applied is Books.
- View: There is a view function called books_list_view, which will use Books.objects.all() to obtain all books in database

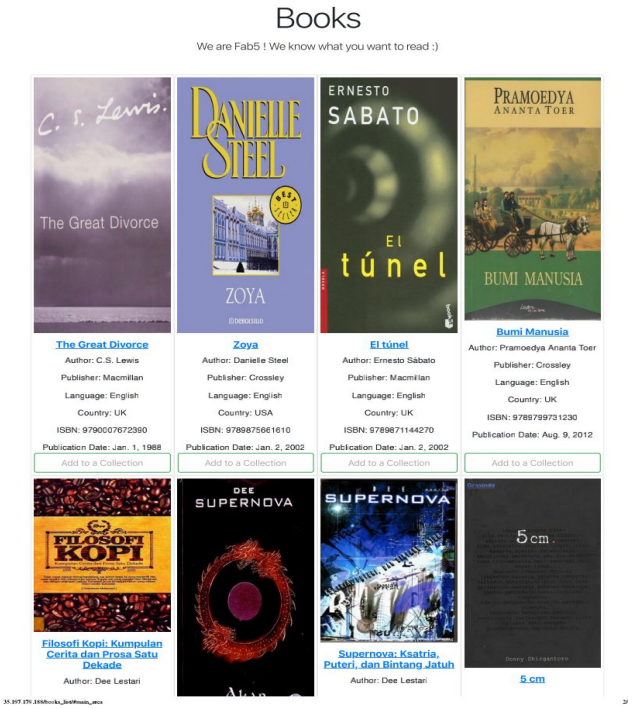
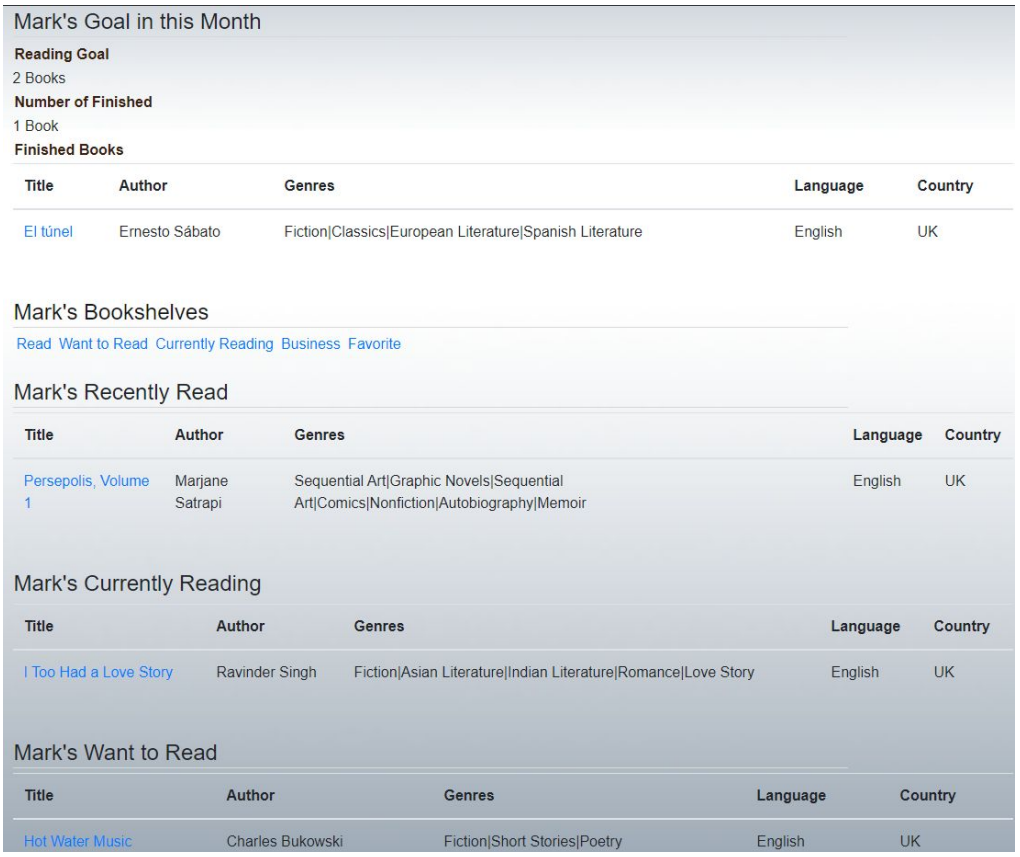


Figure 17. books list page

In this user story, it is not convenient to show all books in one page, so we learn to use Paginator. After applying Paginator, there are 20 books on one page, then the user can click the next page to view other books.

READ-8: As a reader, I want to view books from other users' collections so that I know what other people are reading.



Mark's Goal in this Month

Reading Goal
2 Books
Number of Finished
1 Book
Finished Books

Title	Author	Genres	Language	Country
El túnel	Ernesto Sábato	Fiction|Classics|European Literature|Spanish Literature	English	UK

Mark's Bookshelves

Read Want to Read Currently Reading Business Favorite

Mark's Recently Read

Title	Author	Genres	Language	Country
Persepolis, Volume 1	Marjane Satrapi	Sequential Art|Graphic Novels|Sequential Art|Comics|Nonfiction|Autobiography|Memoir	English	UK

Mark's Currently Reading

Title	Author	Genres	Language	Country
I Too Had a Love Story	Ravinder Singh	Fiction|Asian Literature|Indian Literature|Romance|Love Story	English	UK

Mark's Want to Read

Title	Author	Genres	Language	Country
Hot Water Music	Charles Bukowski	Fiction|Short Stories|Poetry	English	UK

Figure 18. User profile page

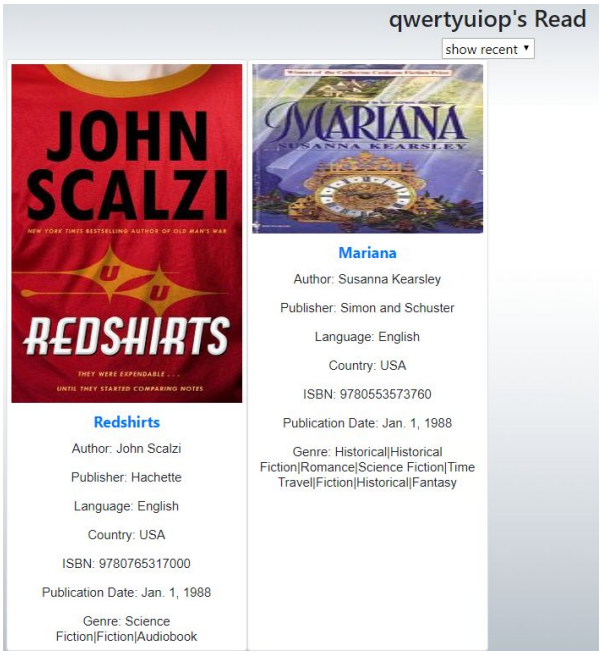


Figure 19. Collection Page

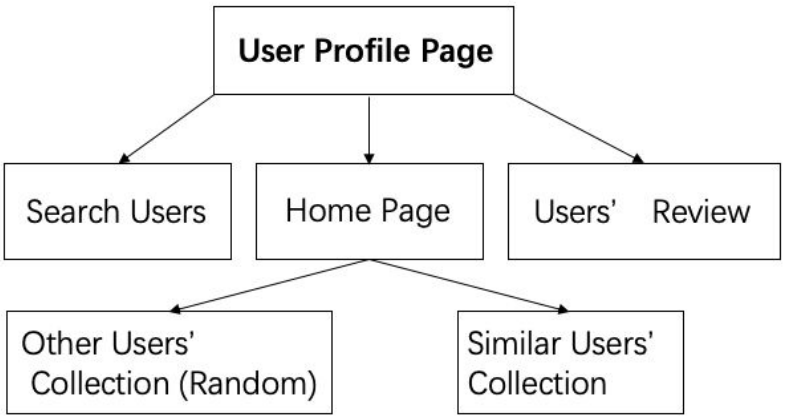


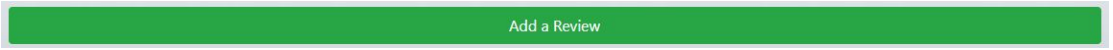
Figure 20. Different ways to access user's collection

There are three default collections under each account, and users also can create their own collections to save books. All users can see others' collections and book details under the collection. We have set an id check function to determine whether the user has permission to edit the collection or not, which means users only can edit their own collections but only can view others' collections. There is a section on the user home page to display other users' collections randomly, and users can access other's collection page from here directly.

Also we have created a user profile page for all users to display the user's reading goal, all collections, and details of three default collections so that users can also access other's collections through the user profile page. We have linked the user profile page with the following situations: 1. search results under user search function; 2. similar users section on user home page; 3. user's review under book page.

There were two issues for this user story. One is that users can edit other user's collections (such as delete books) at the initial stage, which is a very bad experience for users. Then we added a user id check function to determine the user has permission to edit or not. Also, we displayed the user's personal information on the user profile page at the initial stage (such as first name, last name and birthday), but we hid it and changed to display personal goals after considering the personal privacy issues.

READ-9: As a reader, I want to write reviews for a given book, including leaving a rating out of 5 so that other users can know my thoughts about this book.



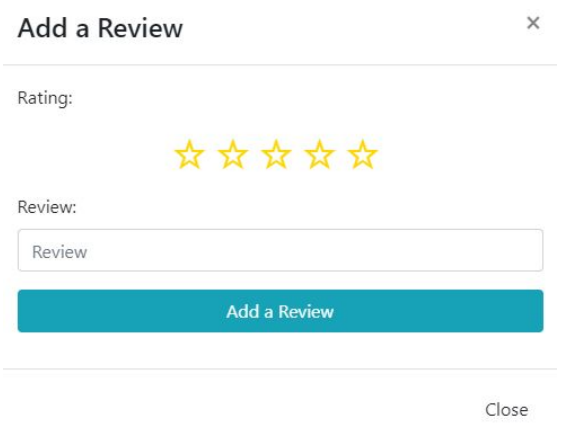


Figure 21 . The template of adding a review

This user story provides the functionality of writing a review for a given book. This maps to the project objective that readers must be able to write reviews for a given book, including leaving a rating out of 5. The reader can click the add a review button on any book page, and choose a rating out of 5 with some writing comment.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Template: a modal form with rating radio buttons and review textbox
- Model: a review model that contains user id, book id, rating number, review text, and createdDate.
- View: There is a view function called add_review, which will accept the POST request. It will check whether the user has added a review before, and insert a review record into the database.

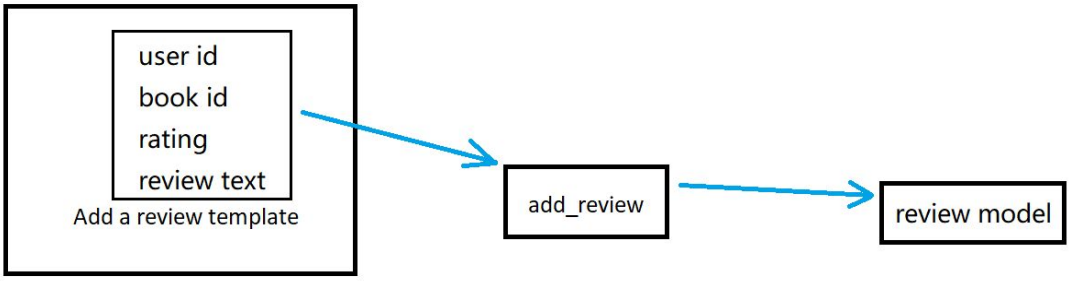


Figure 22 . Adding a review workflow

At this stage, we have learnt about jQuey dynamic. So I decided to use the interactive modal and AJAX to update the book review page. However, during the process, I found the jQuery cannot process the AJAX function. The reason is that we are referencing the slim version of jQuery, which does not contain

the AJAX component (jQuery 2020). After rewriting the reference, the review insertion is working.

READ-10: As a reader, I want to view reviews from other people so that I can know what others think about this book.

Figure 23. The review displaying template

This user story provides the functionality of displaying reviews for a given book. This maps to the project objective that readers must be able to view the full details of any book, (including all reviews). The reader can click the username to view the user profile as well.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Template: a table of three columns: user, rating, review
- Model: a review model that contains user id, book id, rating number, review text, and createdDate.
- View: There is a view function called book_view which will display the review.

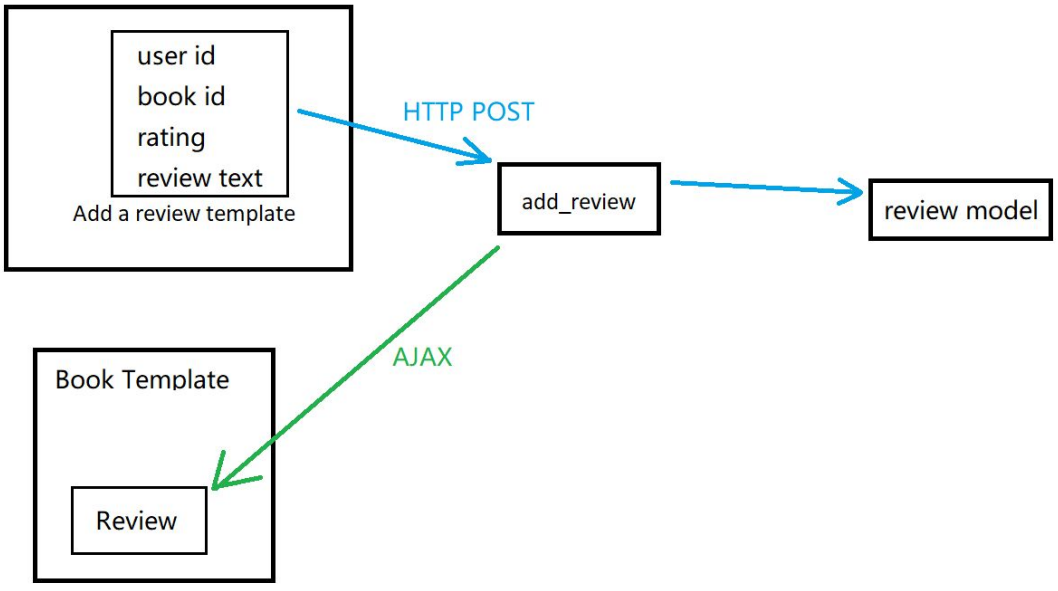


Figure 24. Updating the review section of the book template.

In this stage, we have learnt jQuery techniques to update the review tables asynchronously. During the implementation, I found the updating of the review

table can be done by AJAX after submitting the review. Therefore, without reloading the whole page, we can update the review section separately.

READ-12: As a reader, I want to view the full details of any book (including all reviews and total count of the number of readers), in any book collection so that I can decide whether I would read the book.

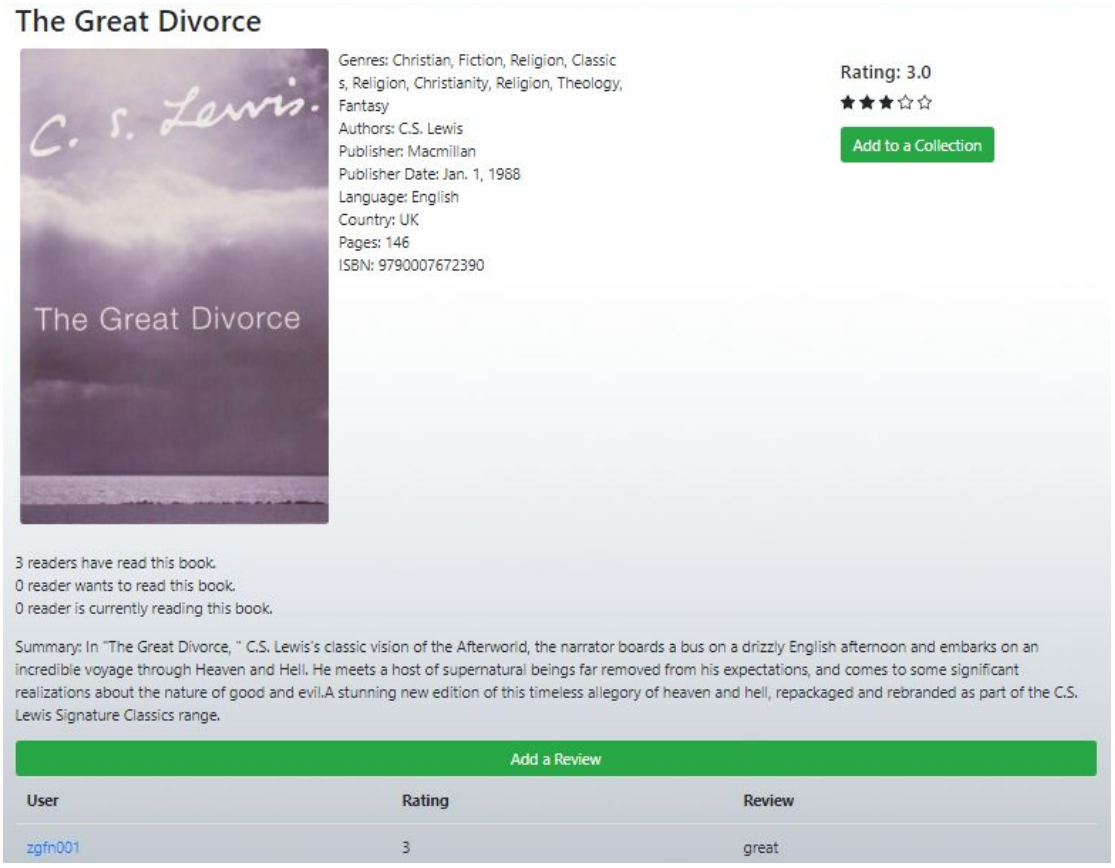


Figure 25. Book Template, includes the detail of the book

This user story provides the functionality of displaying the full details of a book. This maps to the project objective that Readers must be able to view the full details of any book, (including all reviews). Readers can check genres, authors, publisher, publication date, language, country, pages, and ISBN. Readers can also check reviews which have been completed at other user stories.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Template: The book.html contains book cover, attributes such as genres and authors, and a table of reviews.

- Model: the book model contains several fields: ISBN, title, authors, genres, summary, language, country, num_page, cover, publisher, publication_date.
- View: there is a function called book_view which will use the isbn to retrieve the book data and pass them to the template.

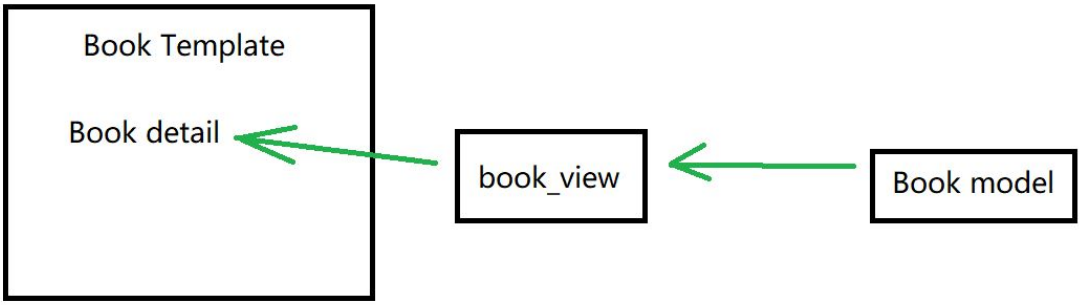


Figure 26 . Book page workflow

This user story is quite straightforward. However, there are many elements in the book.html, which makes it hard to manage. I decided to use snippets to separate each content into a snippet. Django supports snippet reference in order to organize the html. Therefore, book.html has a very clear organization.

READ-13: As a reader, I want to search books by book name, author, countries and languages and filter out books that are under a given average rating so that I can find the book details.

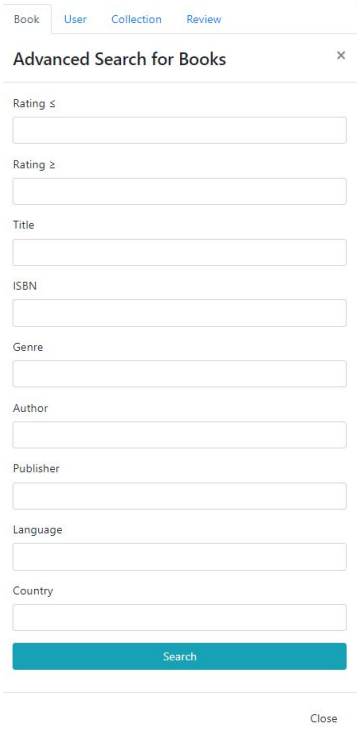


Figure 27. Advanced search book window

This user story provides the functionality of searching a book with given detail. This maps to the project objective that readers must be able to easily search for books by name or author. Readers can search genres, authors, publisher, publication date, language, country, pages, ISBN, and given rating. Readers can also search for users, collections and reviews.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Template: the advanced_search.html would get keywords from readers. The search_result.html would display the result.
- Model: this will use book, account, collection, review models
- View: there are several advanced searching view functions corresponding to each section. The view functions will get key words and use SQL to search from the database.

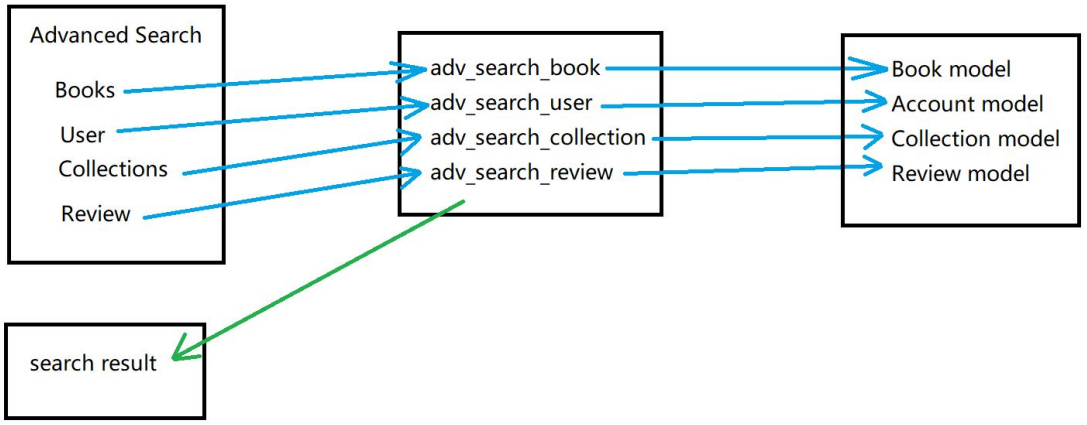


Figure 28. Advanced searching workflow

During the displaying the books, we found if returned too many books, it will acquire too much resource such as the book cover images. Therefore, we decided to use pagination which also makes the book list page and search result page not too long. The loading time is significantly reduced as well.

READ-15: As a reader, I want to get a summary of books I read so that I can determine which kind of book can be read next.

Finished Books

Title	Author	Genres	Language	Country
El túnel	Ernesto Sábato	Fiction|Classics|European Literature|Spanish Literature	English	UK

Figure 29. Finished Books

Mark's Recently Read

Title	Author	Genres	Language	Country
El túnel	Ernesto Sábato	Fiction|Classics|European Literature|Spanish Literature	English	UK
Persepolis, Volume 1	Marjane Satrapi	Sequential Art|Graphic Novels|Sequential Art|Comics|Nonfiction|Autobiography|Memoir	English	UK
Rage of Angels	Sidney Sheldon	Fiction|Thriller|Mystery|Suspense|Romance	English	USA

Figure 30. All Books in Read Collection

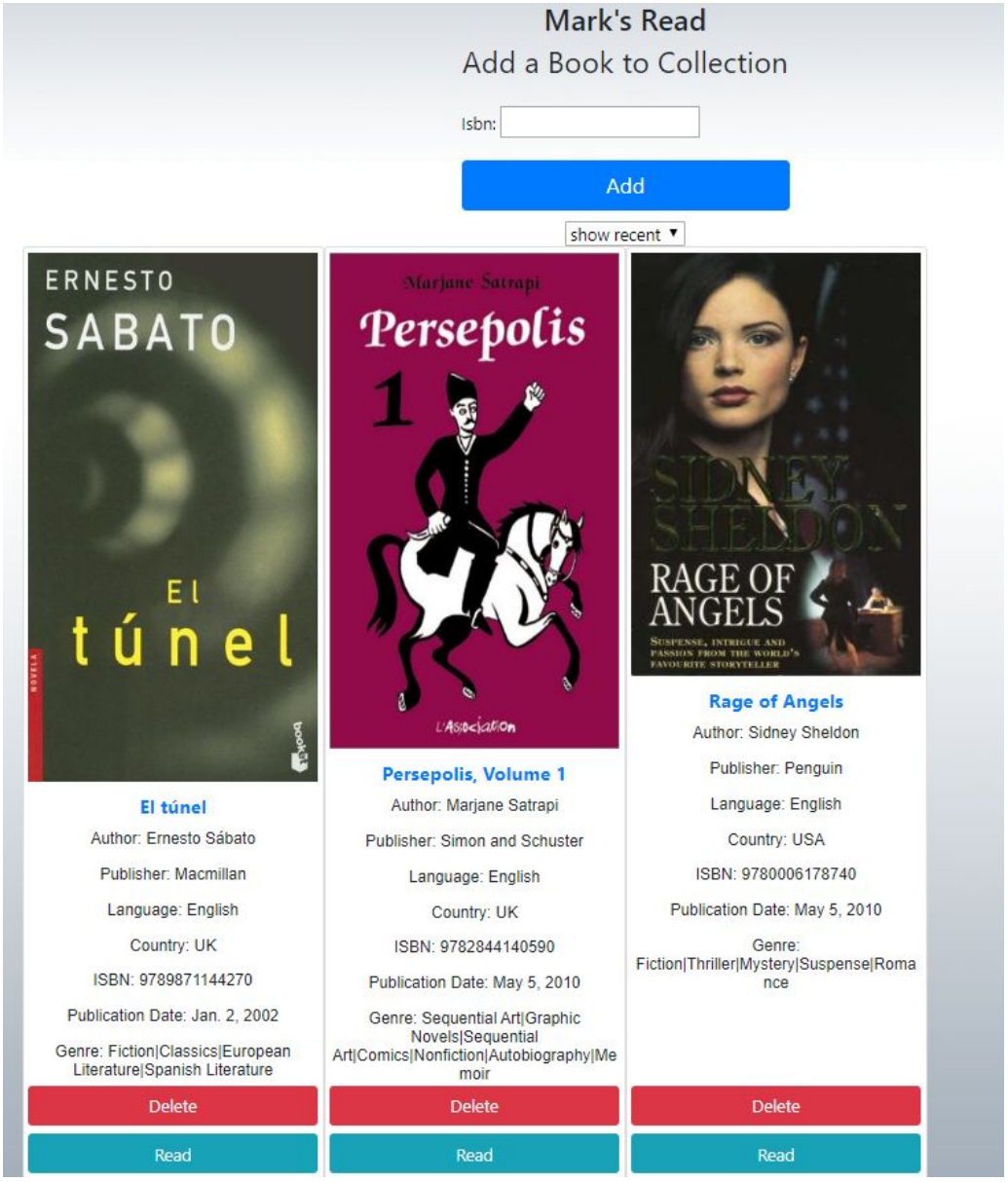


Figure 31. Read Collection Page

There is a button called “My Profile” on the user home page, and all users can enter their profile page from here. Then they will see the summary table of the books they have finished this month, or all books under their “Read” collection. The summary tables include title, author, genres, language, and country of the books, and users can clearly view the summary of books they read.

Before implementing this page, we made the collection page for users and they can see what books they have read in their read collection. This page (Figure 31. Read Collection Page) used the card format to display the book’s details (especially the book cover) which can help the reader recall the book faster. However, this format is not good for readers to have an overview of the

summary of books. Therefore, we created a table format in the user profile page to display the details of books (*Figure 31. All Books in Read Collection)*, and users can easily see the title, author and genre from the table which can help them to know what kind of books they prefer.

READ-16: As a reader, I want to access mine or other users' collections including seeing the recently read top 10 books, so that I can find new books to try.

In this user story, we implemented a show-recent function that readers can choose to reorder books in any collection based on their added datetime. Readers not only can choose to list top 10 books but also 5 or 15 books (*Figure 32.*). When readers select any of their or other users' collections, they can see a show-recent option that can provide them to check the latest added books. By achieving this function, a date time field has been added to an intermediate model which is related to the books field in the collection model. Each time when readers add a book to any of the collection, current datetime will be stored into a database. When readers select any number in show recent option, it will trigger a JQuery function that sends an Ajax request to showRecent view with the collection id and the selected value. In showRecent view, books in that collection will be ordered by added date time decreasingly and select top that number and sent back to html page with related data. When the response is successfully received, current html tags will be removed and rewritten based on the data received.

The hardest part of showing the recent books is how to reorder books without reloading the current page. In order to do that, we use JQuery and Ajax to first get json data like collection id and send back to the view function to order books and respond with those reordered books. When Ajax successfully receives the response, it will rewrite the html only for the displaying-books part.

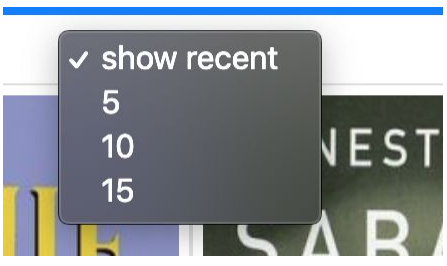


Figure 32. Show Recent

READ-34: As a reader, I want to browse some random books so that I can explore unknown books.

The functionality which READ-34 provides is to show some random books on the user homepage. The aim for the user story is to provide more possible interesting books to users. Users can click the book title to look through the detail of a book and the displaying books will be updated when the user homepage is refreshed.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Model: the model applied is Books.
- View: The view applied is user_homepage_view, where Books.objects.all() will find all books in the database then sample() is applied to obtain 4 random books in all books.
- Template: the template applied is user_homepage.html, where the content container card is used to show cover, title, and author of each book.

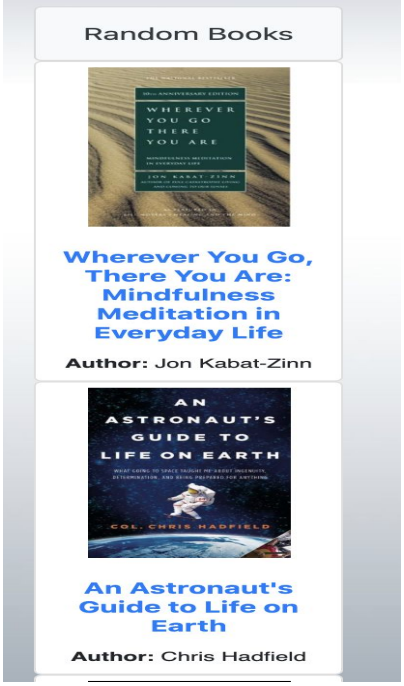


Figure 33 . The random books showed on the user home page.

READ-35: As a reader, I want to browse some random collections so that I can explore unknown books.

The functionality which READ-35 provides is to show users' own collections and other users' random collections on the user homepage. The objective which the user story satisfied is " They must be able to look through any such collection (their own and that of other readers)". The user can click the collection name to look through all books in this collection.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Model: the model applied is Collection and Account.
- View: the view applied is user_homepage_view. On one hand, Collection.objects.filter() is applied to find user own collections. On the other hand, Account.objects.exclude() will find the other users in the database except user own, then Collection.objects.filter() is used to find all the other users' collections, then sample() will choose 4 collections randomly in these collections.
- Template: the template applied is user_homepage.html, where the content container card is used to show the user's own collections and 4 other users' collections.

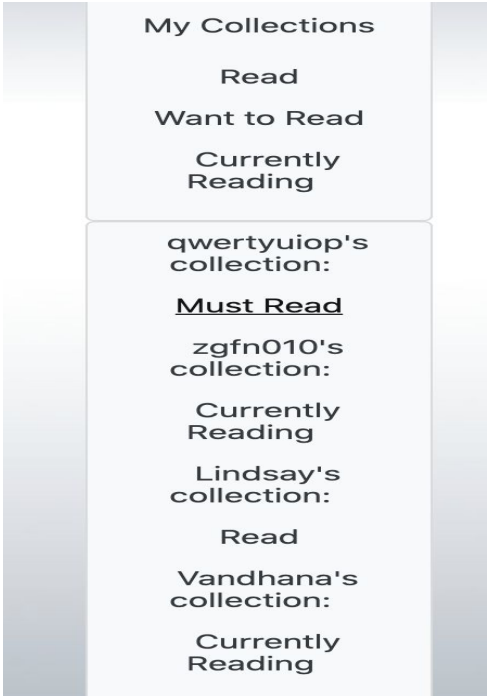


Figure 34 . The user own and other users' random collections showed on the user home page.

The challenge for this user story is how to design a block to show the collection. I used bootstrap .well class at first, but there is no gray background shown, so i try many methods to figure it out. Finally, I found that the well class has been removed after the 4.0 version of bootstrap. But i found a more powerful class to institute for .well class, it is card class in bootstrap.

READ-108: As a reader, I want to delete and edit my collections so that I can be more flexible with keeping my collections

After readers create their named collections, readers can choose to edit their own collection name or descriptions and delete the collection. System default collections like “Read”, “Currently Reading” and “Want to Read” are set to unmodifiable (*Figure 35.*). When readers change the collection name to an existing one of their collections, the system will reject this change and give readers a notice (*Figure 36.*). After readers delete a collection, the related books in that collection will also be deleted.

The difficult part in this user story is how to forbid readers to make modifications on default collections (Read, Want to Read, Currently Reading). One way to solve it is to add an if statement in current html page which is if the collection name is one of the default collections, edit and delete button would not be shown to readers.

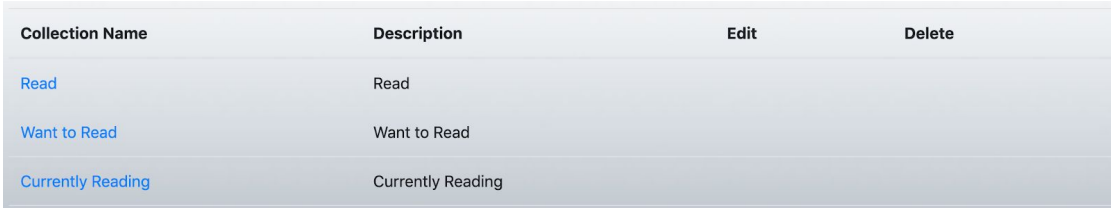


Figure 35 . Collections list

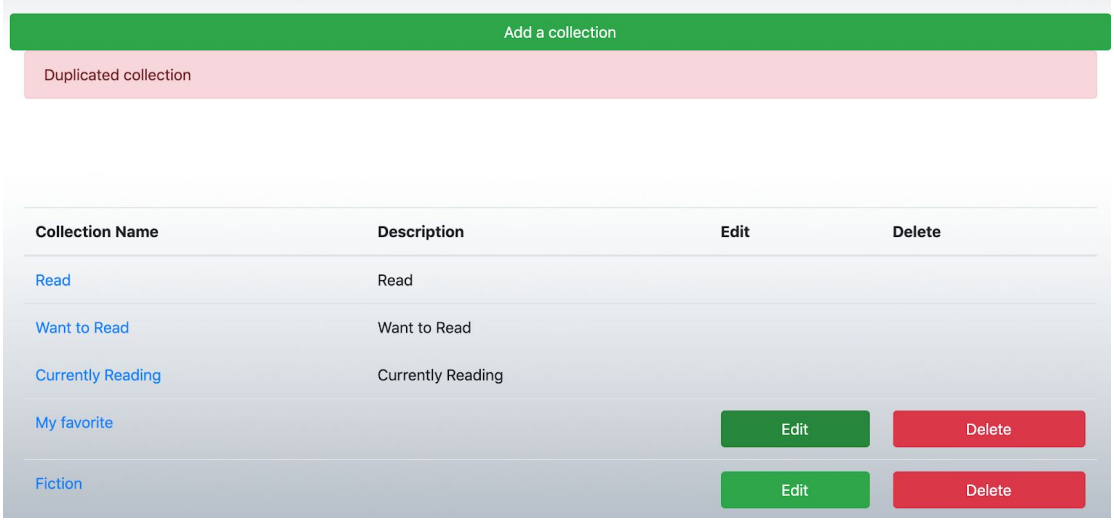


Figure 36. Detect an error for duplicated collection

Sprint 3

READ-17: As a reader, I want to be able to view aggregate statistics, including the average rating of the book so I know if the book is actually good.

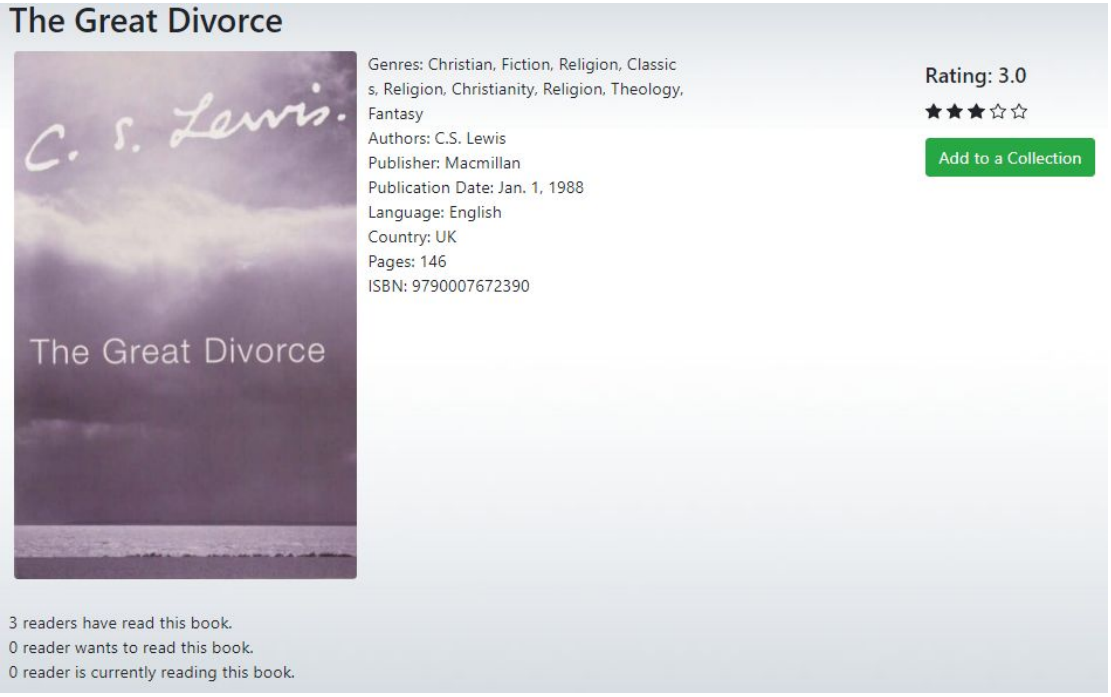


Figure 37. The aggregate statistics about how many readers read the book, and the average rating of the book.

This user story provides the functionality of showing the statistics of how many readers read the book, and the average rating of the book. This maps to the project objective that readers must be able to see the full details view which must display multiple aggregate statistics, including the book's average rating. Readers can use the readers number and average rating to evaluate whether they want to read the book.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Template: Book template contains the readers aggregate statistics and average rating.
- Model: Book model, Collection model, Review model
- View: Book view function, which contains the aggregation function for the number of readers and average rating

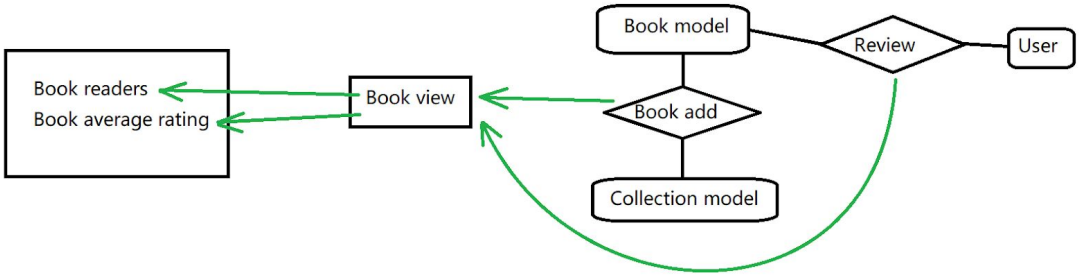


Figure 38 . Book readers statistics and average rating workflow

Surprisingly, the aggregation is not the hard part of this user story. The real hard part is how to display the average rating. Since python will round the decimal point differently for 0.5, 1.5, 2.5 and so on, I decided to use ceiling and flooring to calculate the average rating. Then, using svg with fill star, half fill star and unfill star with corresponding numbers from the average rating, I am able to display the average rating with partial stars.

READ-18: As a reader, I want to be able to set a goal to read a certain number of books every month, based on my collections so I can read everything in my collection.

In this user story, readers can set a monthly goal about how many books they want to read (*Figure 40.*). After readers finish a book during the current month, the system will record the book and record the progress of reading. Readers can find a button “read” in their collection under each book they add into. When readers finish reading a book, they can click the “read” button which will trigger the system to record the progress and the detail of that book.

If readers want to check the detail of their goal, they can go to the user page and select “view challenge” which will direct to the goal detail page (*Figure 39.*). In the goal detail page, readers can choose to delete and edit the monthly goal (*Figure 39.*). At the same time, books that have been read will also be listed under this page.

The most difficult part in this user story is how to record the books readers have read during this month. We add another button “Read” under each book in collections. As long as readers set a goal, the “Read” button will appear. By clicking this button, the system will add 1 to the number of books readers have been finished reading and that book will also be added.

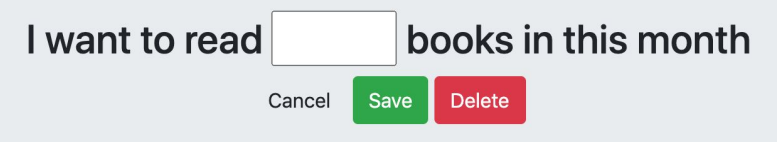


Figure 39. Editing and delete goal

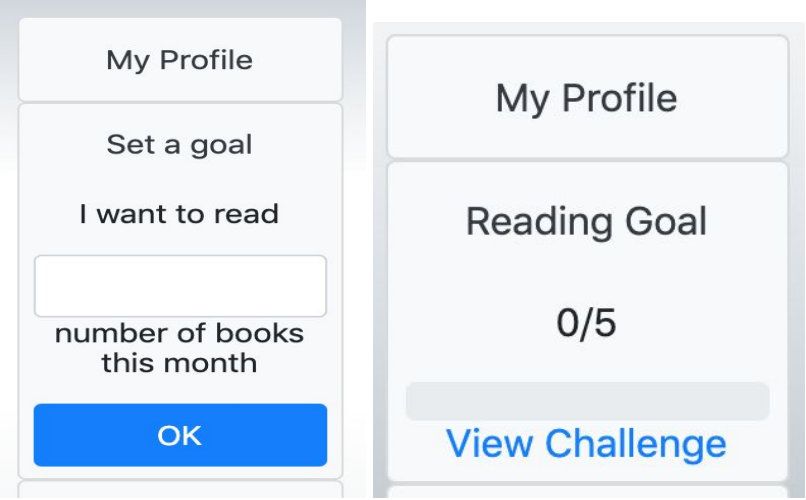


Figure 40. User reading goal setting and viewing

READ-21: As a reader, I want to view the recommendations by selecting a book and a recommendation mode so that I can easily find the books that I am interested in.

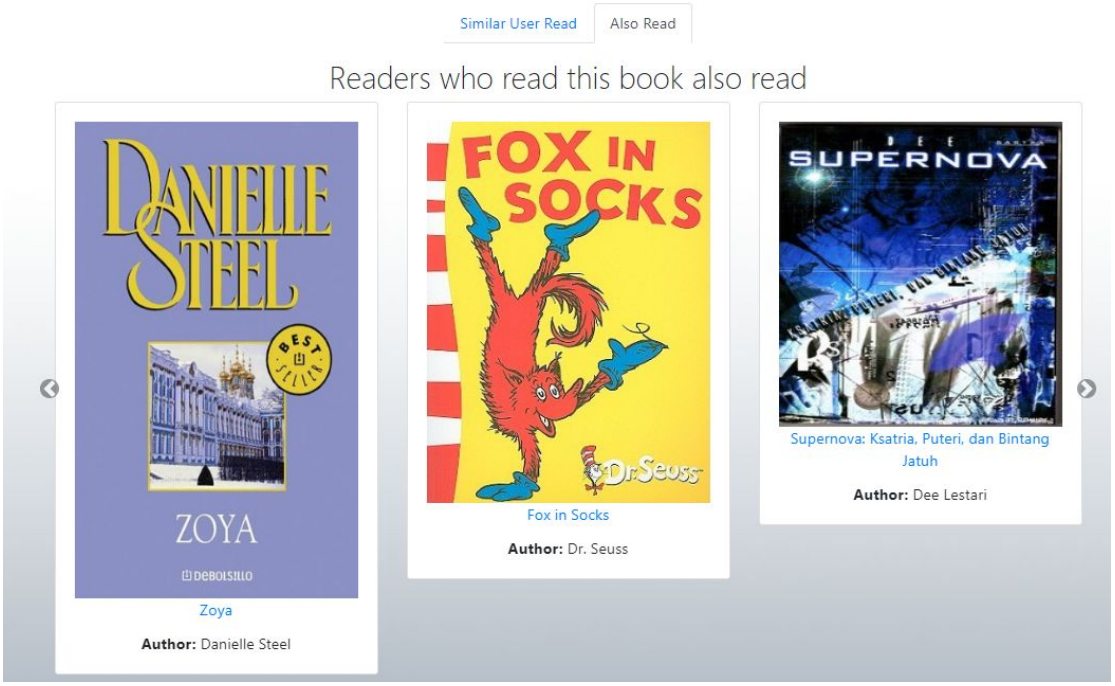


Figure 41. recommend book with "also read" and "similar user read"

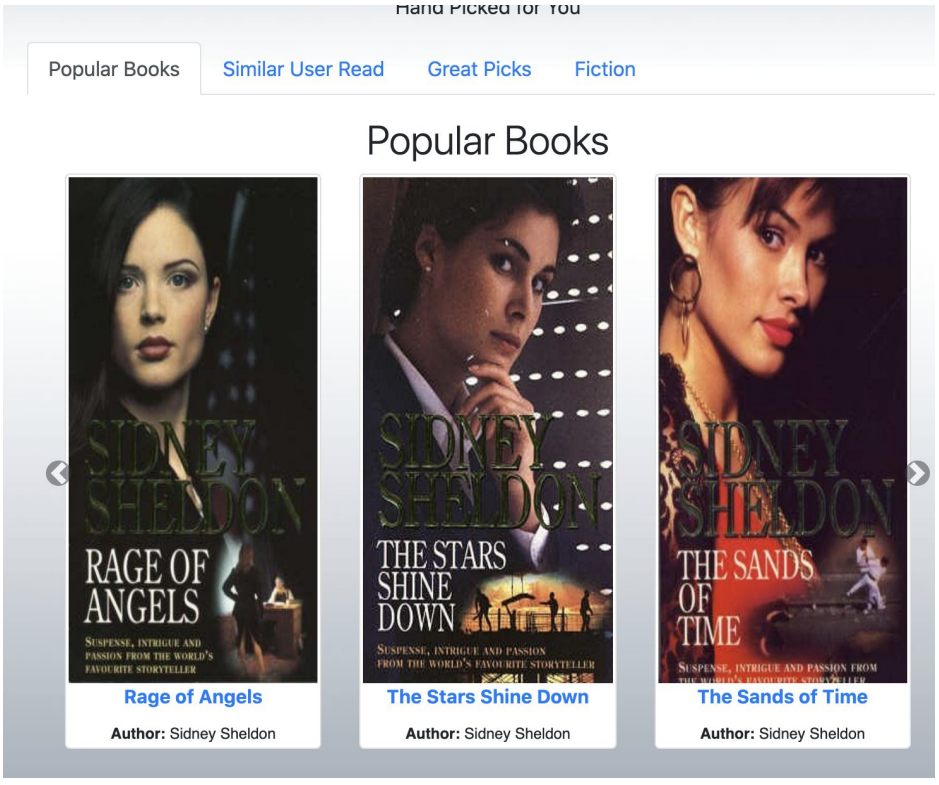


Figure 42. recommend book with “popular books”

This user story provides the functionality of recommending the books read by other users. This maps to the project objective that given a book specified by the reader, the platform must also be able to provide the reader with multiple recommendation modes that can be used to recommend a book. The “Also read” section would recommend the books read by other readers who have read this book. The “similar user read” section would recommend the books read by similar readers as you based on user profile. The “popular books” section would recommend a book if a book is added in more collections.

This function is achieved by using Model–template–view (MTV) software design pattern.

- Template: a carousel card to displaying books
- Model: book model
- View: book_view which will provides the recommend books list

This “popular books” recommendation mode function is achieved by using Model–template–view (MTV) software design pattern.

- Template: a carousel card to displaying books
- Model: Books model and Collection model.
- View: user_homepage_view which will calculate the number of collections where each book is added, then the books which are in more collections will be added.

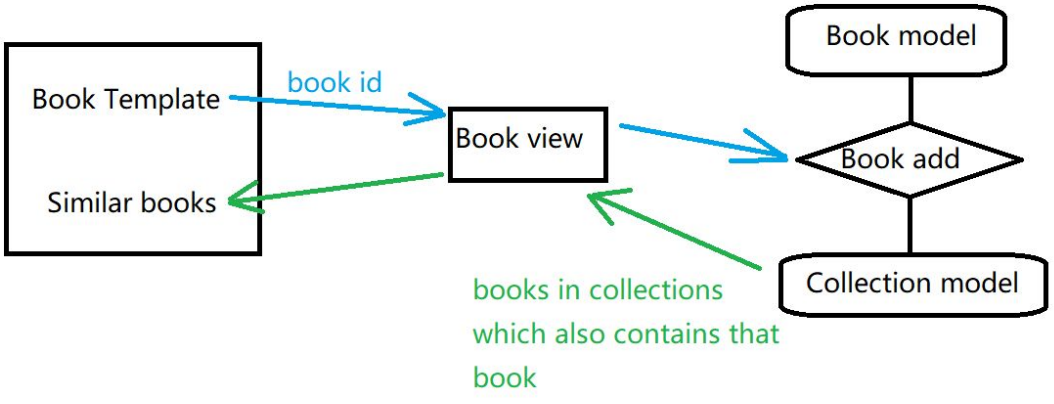


Figure 43. Recommendation workflow for “Also read”

The view function is very easy, by using SQL, it can find the collection which contains the same books, then randomly choose some books to display on the book recommendation template. However, the displaying recommendation

section is not an easy job. The carousel class can only display one item at a time, which is like a PPT slide. Therefore, in order to display more books, I used css to set each item as a part of the displaying item. Then, each time it moves left or right, the jQuery function will insert and remove an item from the active item.

READ-61: As a reader, I want to read books that are popular for my age/gender/occupation so that I can find my interests.

There is a field called “similar users” under the account table, which means all similar users will be saved in here for each user. If the current user is the only user in the database, then he/she has no similar users. If there are less than 5 users in the database, the similar users will be all of the other users. But if there are more than 5 users in the database, the similar users function will be triggered. This function will use gender and birthday as features and use k-nearest neighborhoods algorithm to calculate the user similarity matrix, and update the top 4 similar users to each account.

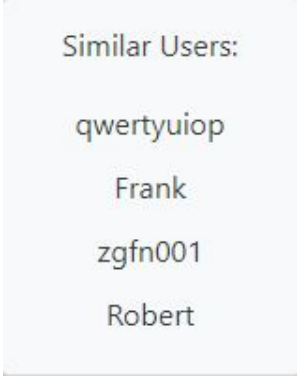


Figure 44. Similar Users Collection

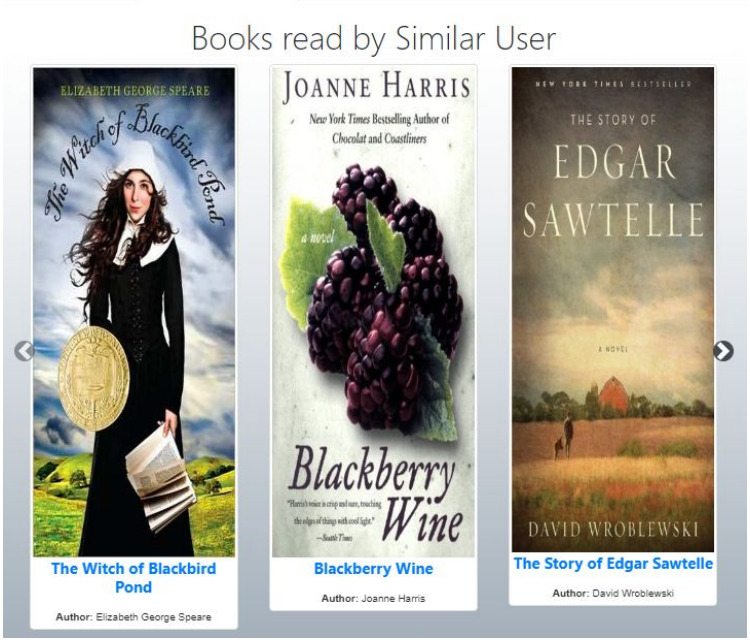


Figure 45. Books read by Similar Users

Users can see their similar users on their user home page, and also can see the books read by similar users under the “Similar user read” mode on both home page and book page.

At the initial stage, we didn’t create the similar user field for the account so that we have to calculate the similar user every time when the user opens the user home page or book page, and it will result in a long loading time. For more efficient purposes, we decided to add the similar users field to record the similar users, and it will be updated when a new user registered. Then we can use it directly on the home page and book page and no need to calculate it every time.

The major challenges during implementation include:

1. Overcoming the learning curve in a short period of time.
2. Getting accustomed to the Google Cloud Platform
3. Using SCRUM and learning the terminologies for the new methodology.
4. Using JIRA to keep track of tasks and GIT to manage code.

Engineering Practices

The team made use of an AGILE method called SCRUM to manage the project. The project was implemented over a series of three sprints where each sprint lasted two - three weeks. JIRA was used to manage task division and keep track of the team's progress and the individual progress of each team member.

After every sprint, the scrum burn down chart and cumulative workflow diagrams were used to monitor the team's efficiency and learn how to improve performance for subsequent sprints.

Retrospective meetings after every sprint helped the team stay focused and learn what went well, what could be improved and what we could do as a team to make this process a success.

Taking responsibility for tasks in subsequent sprints to minimise faults helped the team realise how there is room for improvement.

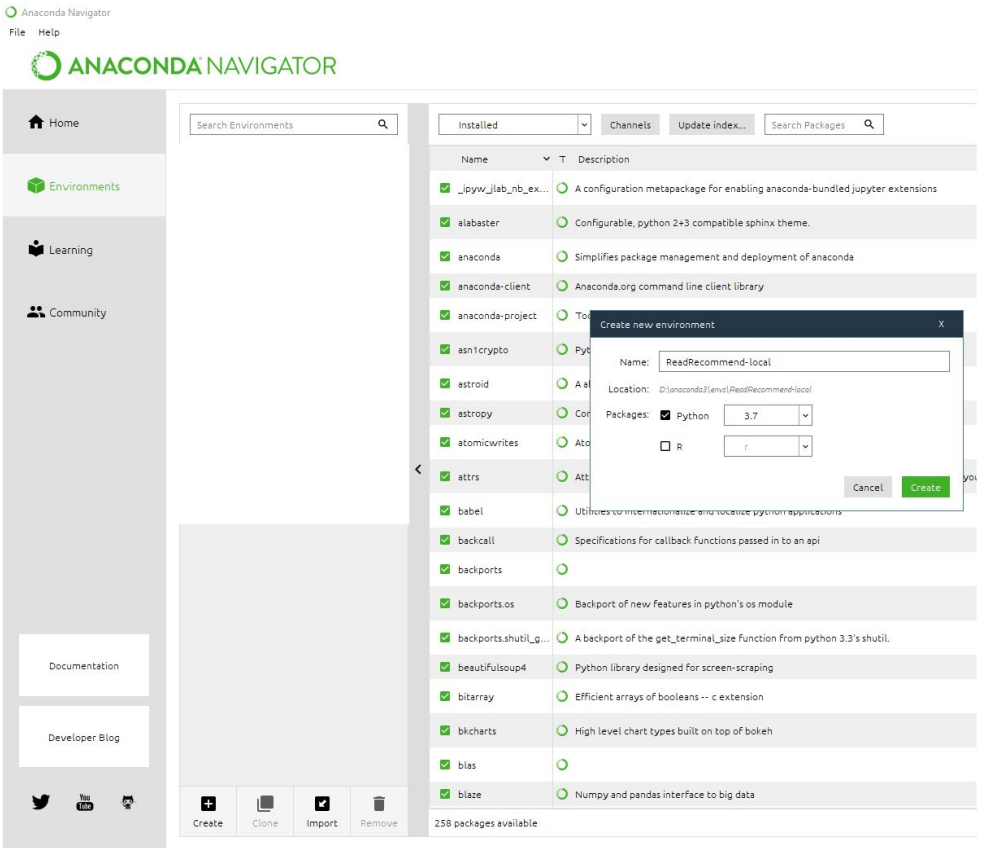
User Document/Manual

Local development

Local: Create python environment

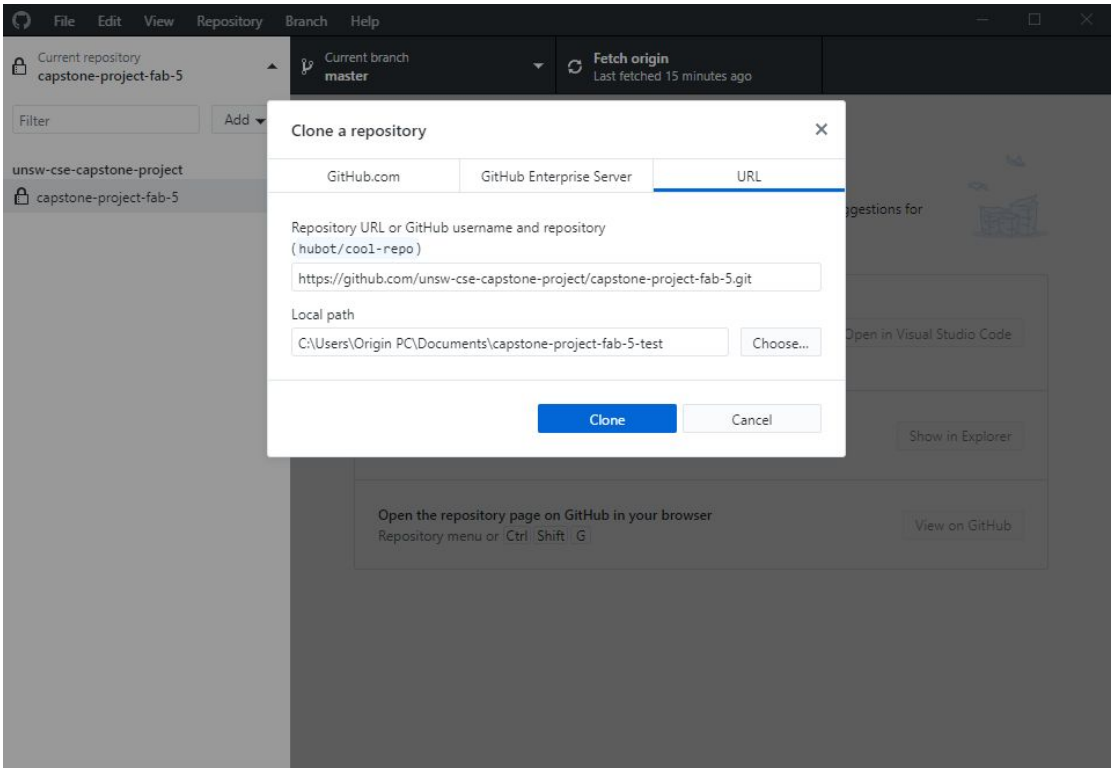
1. Download anaconda, https://www.anaconda.com/products/individual

2. Create an environment

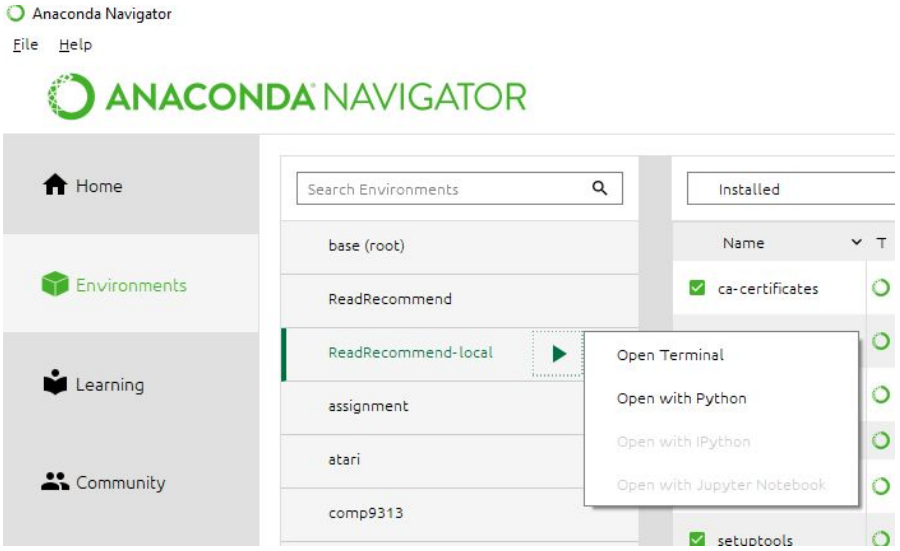


3. Download github desktop https://desktop.github.com/

4. Clone the repository



5. Open a terminal with the environment



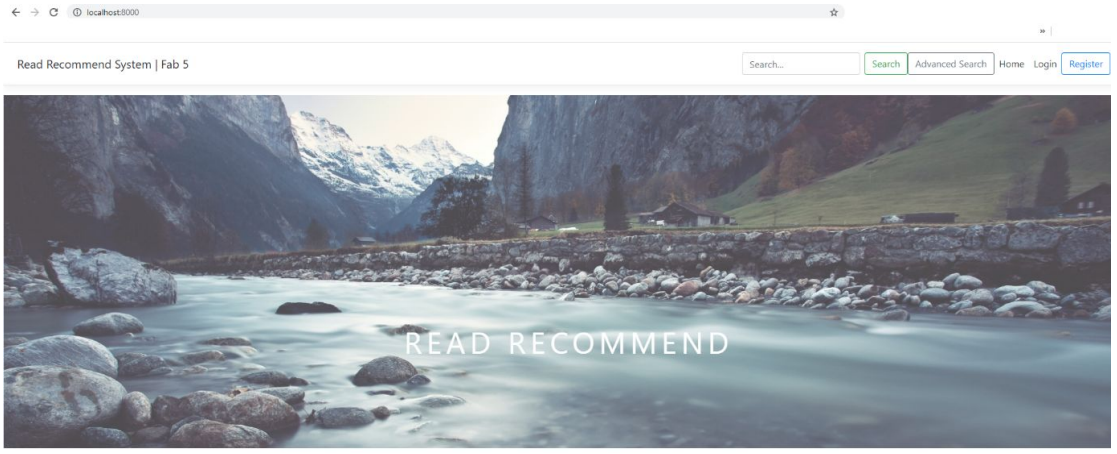
6. Go to the repository
7. $ pip install -r requirements_local.txt
8. Migrate the models

```
(ReadRecommend-local) C:\Users\Origin PC\Documents\capstone-project-fab-5-test>cd mysite

(ReadRecommend-local) C:\Users\Origin PC\Documents\capstone-project-fab-5-test\mysite>python manage.py migrate
Are you on localhost? (y/n): y
Whatever to log
Operations to perform:
  Apply all migrations: account, admin, auth, books, contenttypes, review, sessions, usercollections
Running migrations:
  Applying account.0001_initial... OK
  Applying contenttypes.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying books.0001_initial... OK
  Applying books.0002_auto_20200629_0150... OK
  Applying books.0003_auto_20200714_2324... OK
  Applying review.0001_initial... OK
  Applying review.0002_review_createddate... OK
  Applying sessions.0001_initial... OK
  Applying usercollections.0001_initial... OK
  Applying usercollections.0002_auto_20200701_0015... OK
  Applying usercollections.0003_auto_20200701_0059... OK
  Applying usercollections.0004_auto_20200701_0113... OK
  Applying usercollections.0005_booksadd... OK
  Applying usercollections.0006_auto_20200708_0601... OK
  Applying usercollections.0007_remove_collection_books... OK
  Applying usercollections.0008_collection_books... OK

(ReadRecommend-local) C:\Users\Origin PC\Documents\capstone-project-fab-5-test\mysite>
```

9. Create a folder called media in capstone-project-fab-5-test\mysite
10. $ python manage.py collectstatic

```
(ReadRecommend-local) C:\Users\Origin PC\Documents\capstone-project-fab-5-test\mysite>python manage.py collectstatic
Are you on localhost? (y/n): y

You have requested to collect static files at the destination
location as specified in your settings:

    C:\Users\Origin PC\Documents\capstone-project-fab-5-test\mysite\static

This will overwrite existing files!
Are you sure you want to do this?

Type 'yes' to continue, or 'no' to cancel: yes

130 static files copied to 'C:\Users\Origin PC\Documents\capstone-project-fab-5-test\mysite\static'.

(ReadRecommend-local) C:\Users\Origin PC\Documents\capstone-project-fab-5-test\mysite>
```

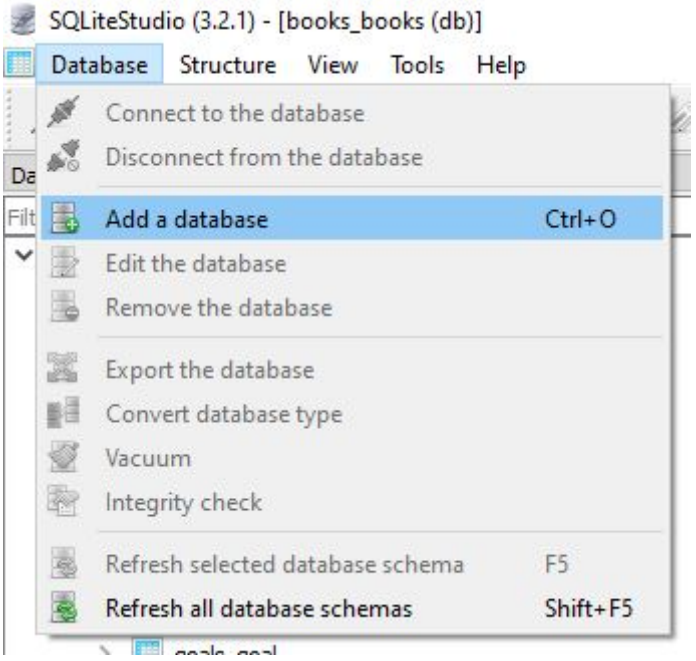
11. Test server, $ python manage.py runserver



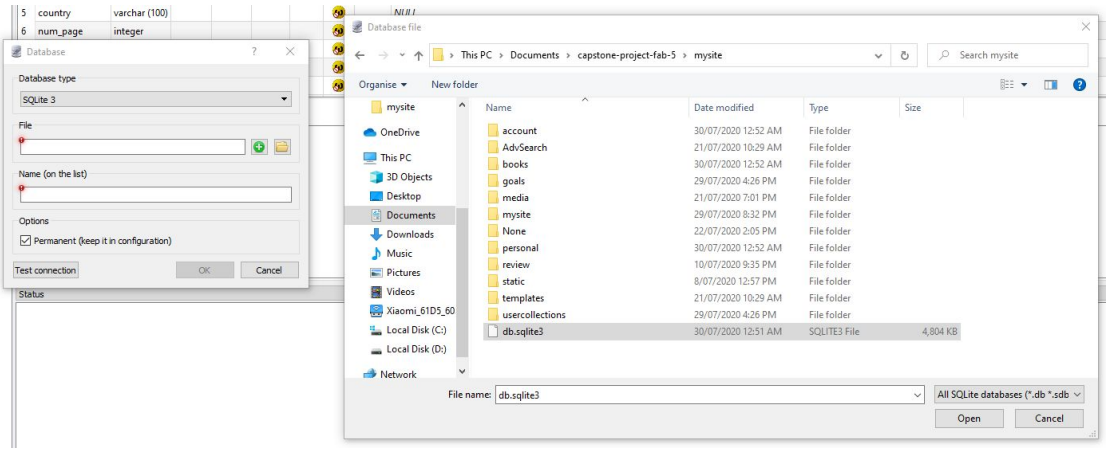
Local: Insert book data

1. Ensure you have finished the above steps, especially the migrate step.
2. Download sqlite studio: https://sqlitestudio.pl/

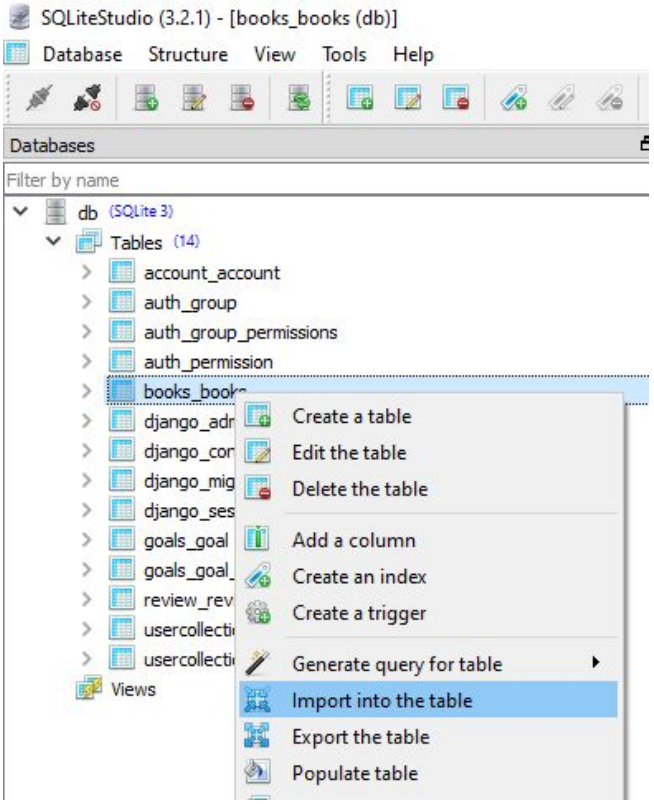
3. Add a database



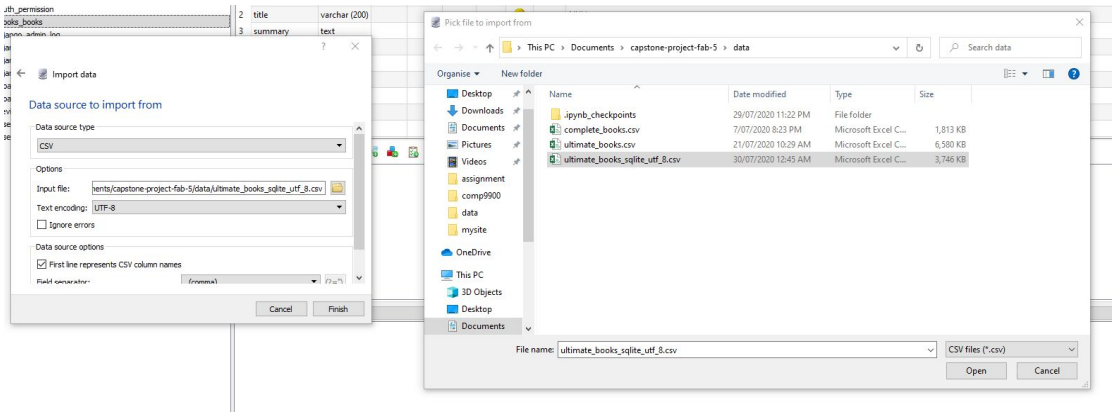
4. Choose dq3.sqlite file from mysite directory



5. Import data from csv



6. Choose ultimate_books_sqlite_utf_8.csv from the data directory

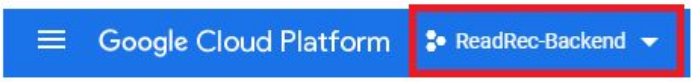


7. Import with utf-8

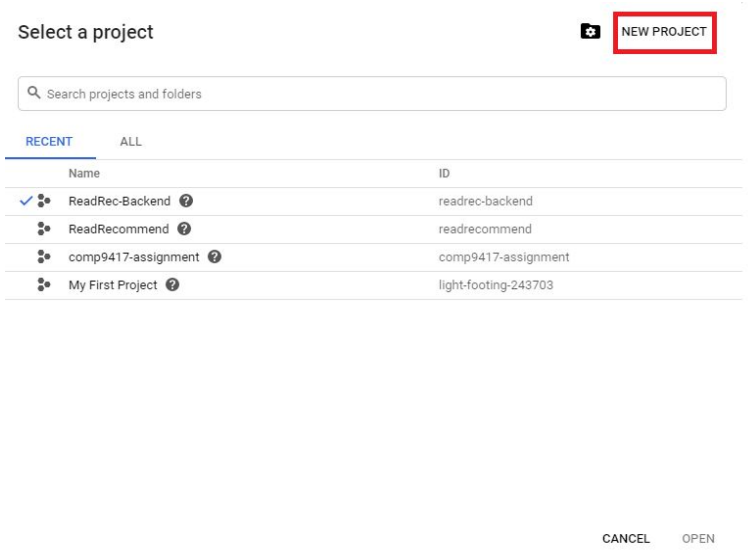
Cloud deployment

Cloud: Create a Google cloud project

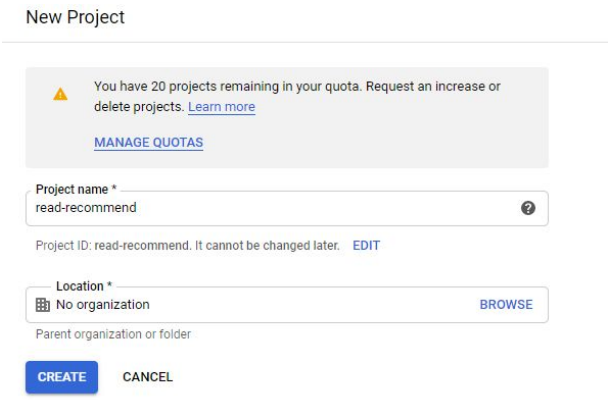
1. Click the project drop down list.



2. Create a new project



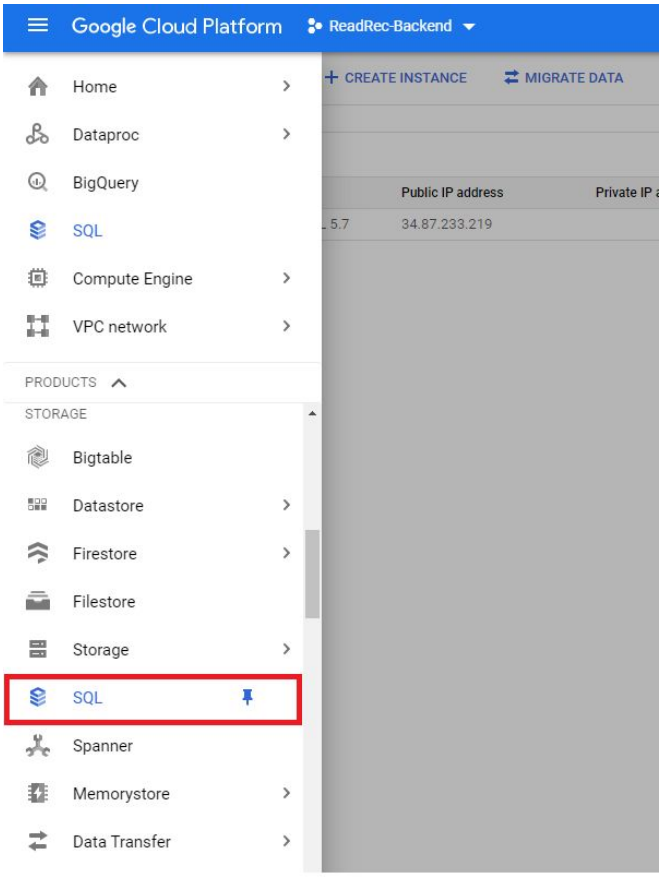
3. Enter the project name and location



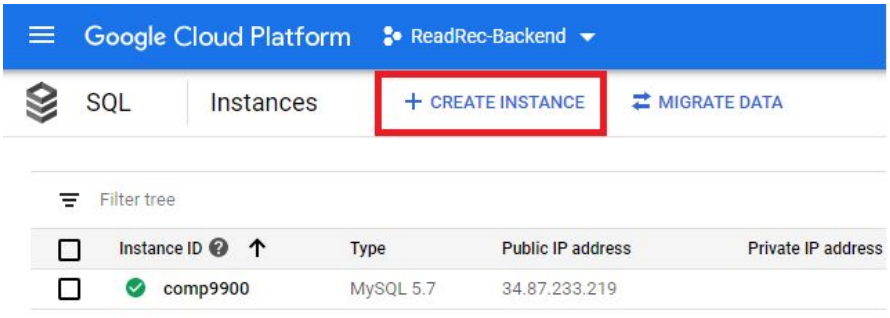
4. Click create button

Cloud: Create a Database Instance

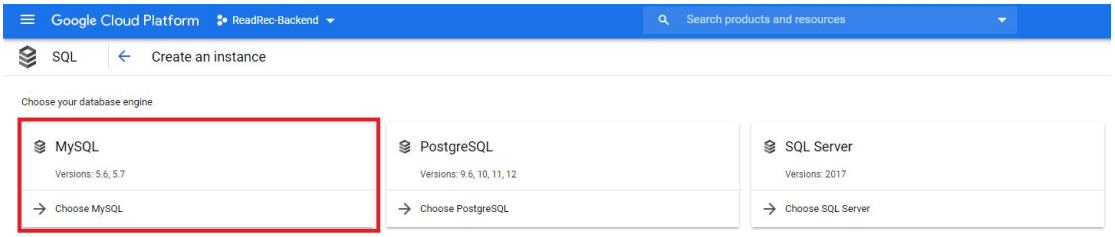
1. Go to SQL dashboard



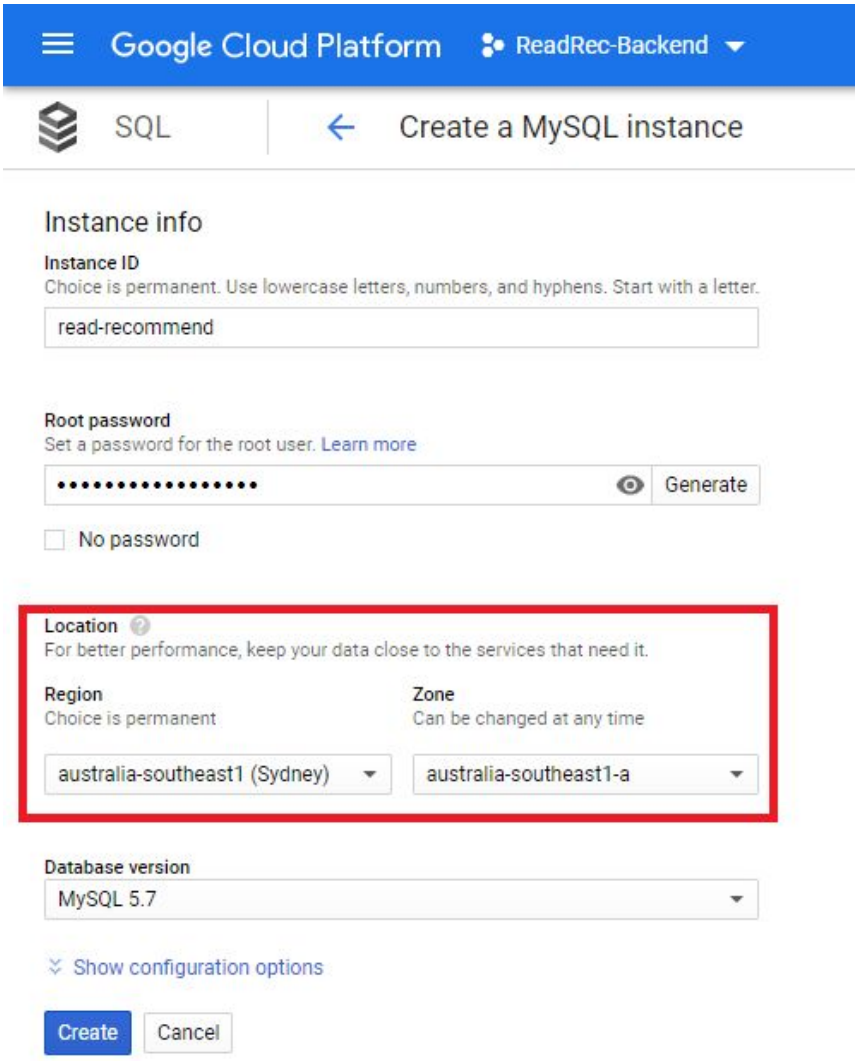
2. Click Create Instance



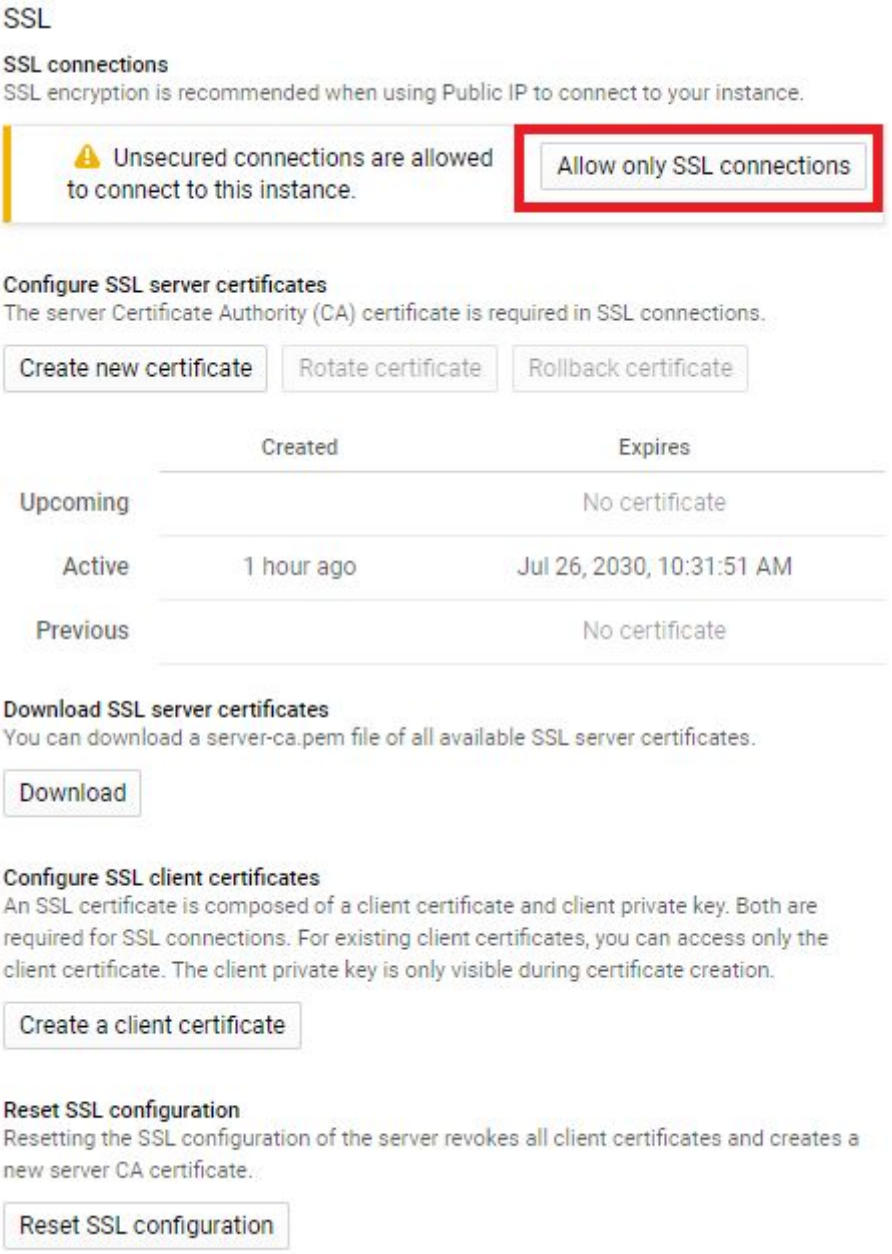
3. Click MySQL



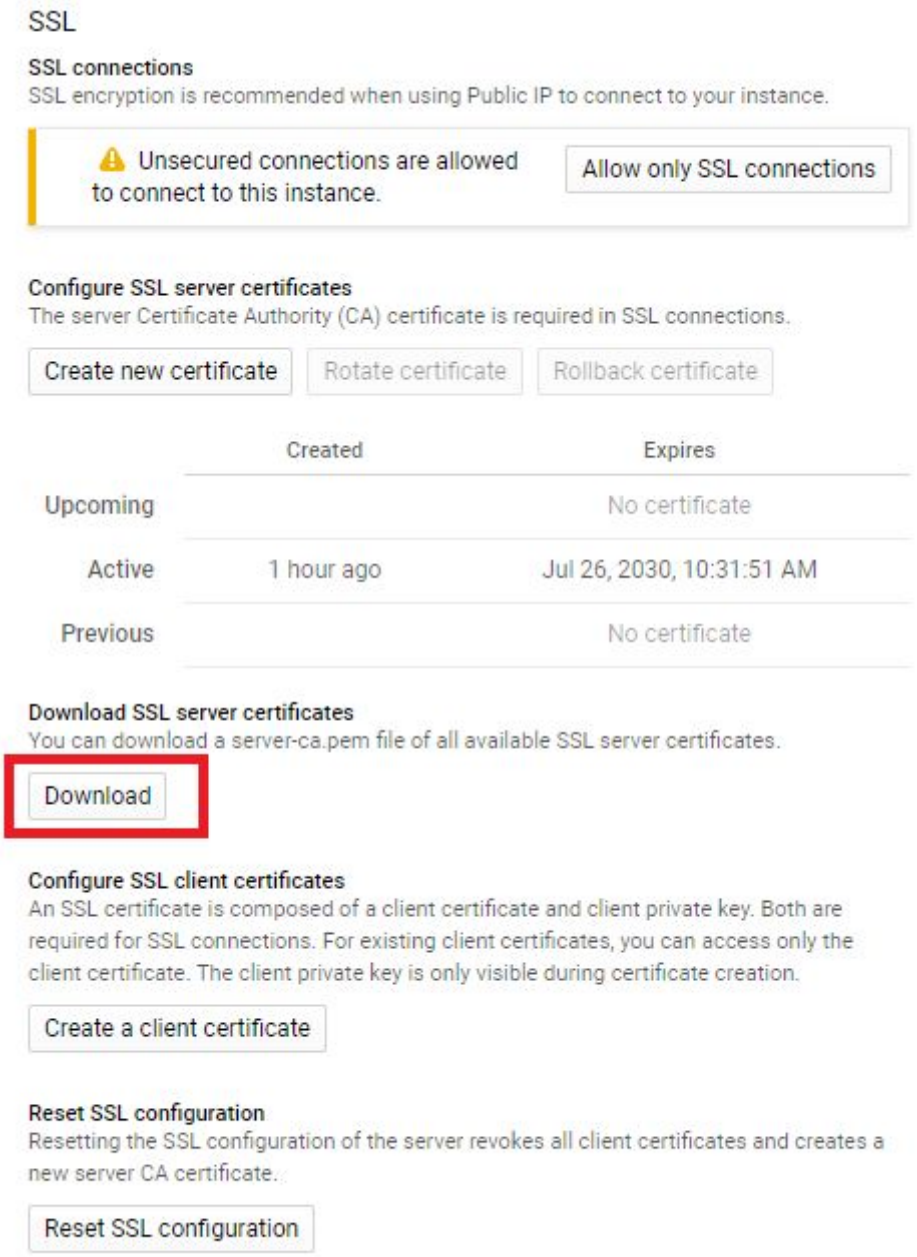
4. Write the details of the database instance. Choose Australia-southeast1(Sydney), australia-southeast1-a. If you choose a different region and zone, please make sure it is the same as the vm instance.



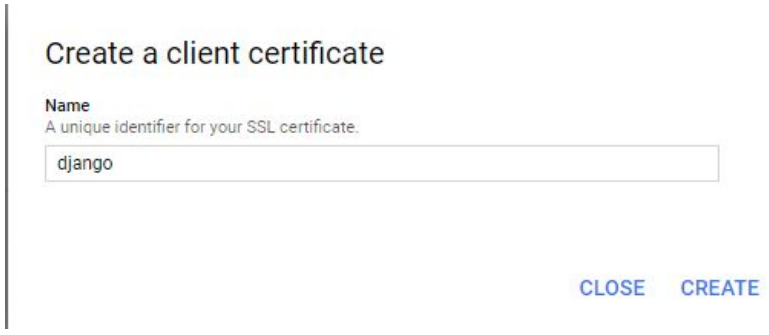
5. Click create button
6. Configure connections, click allow only SSL connections



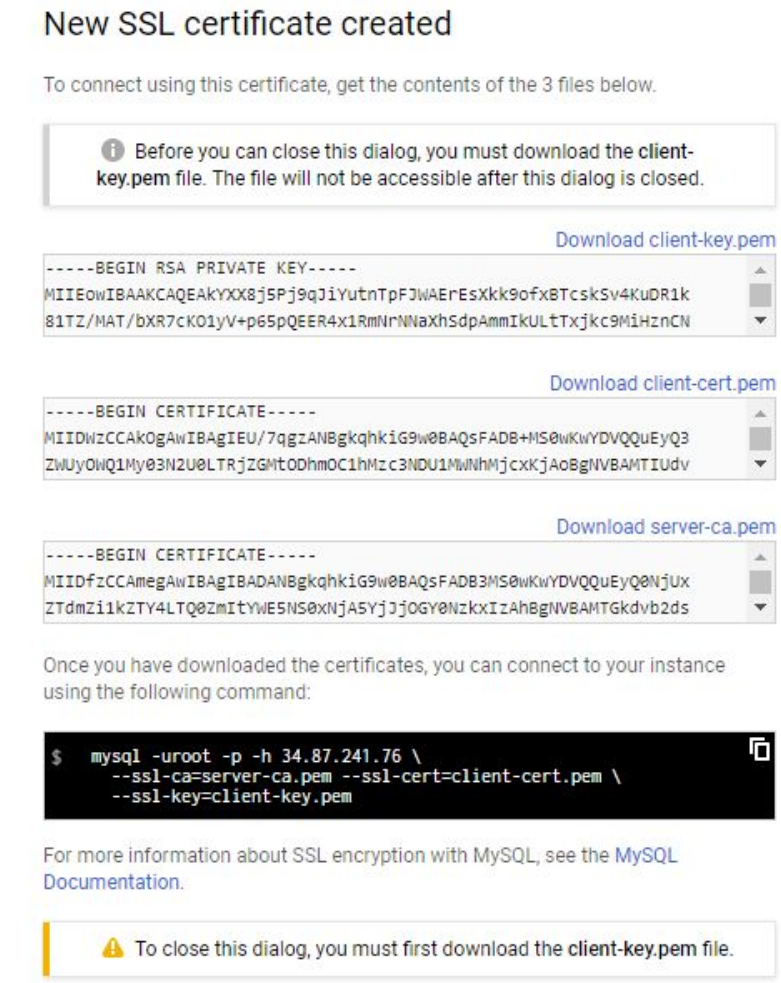
7. Download server-ca.pem



8. Create a client certificate

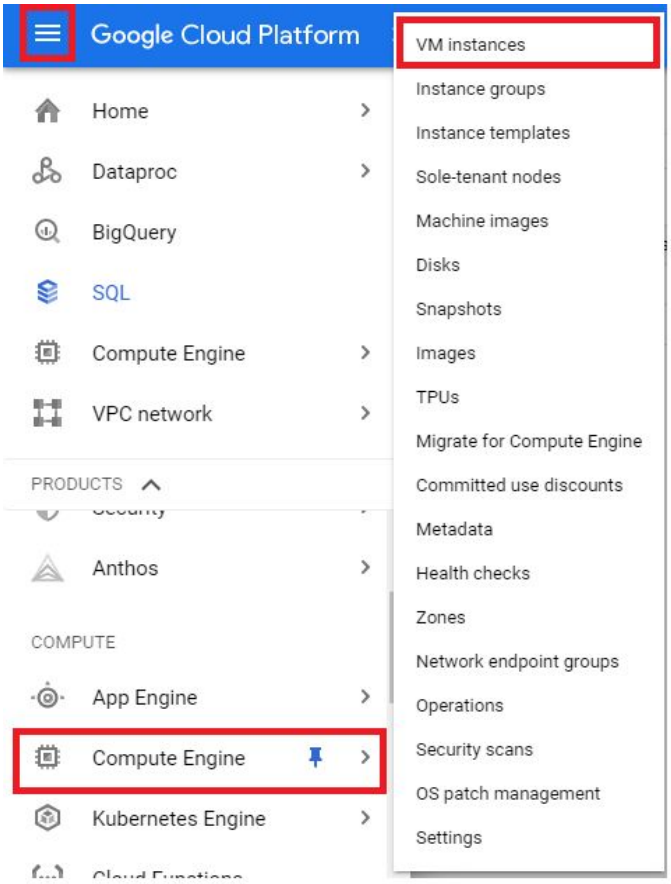


9. Download client-key.pem, client-cert.pem, server-ca.pem, replace the capstone directory's old certificates.

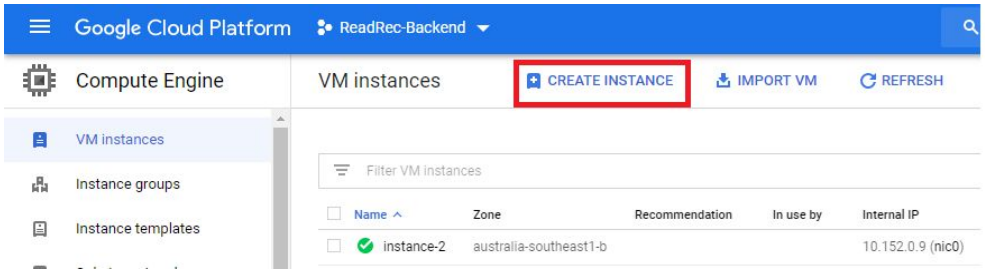


Cloud: Create a vm instance

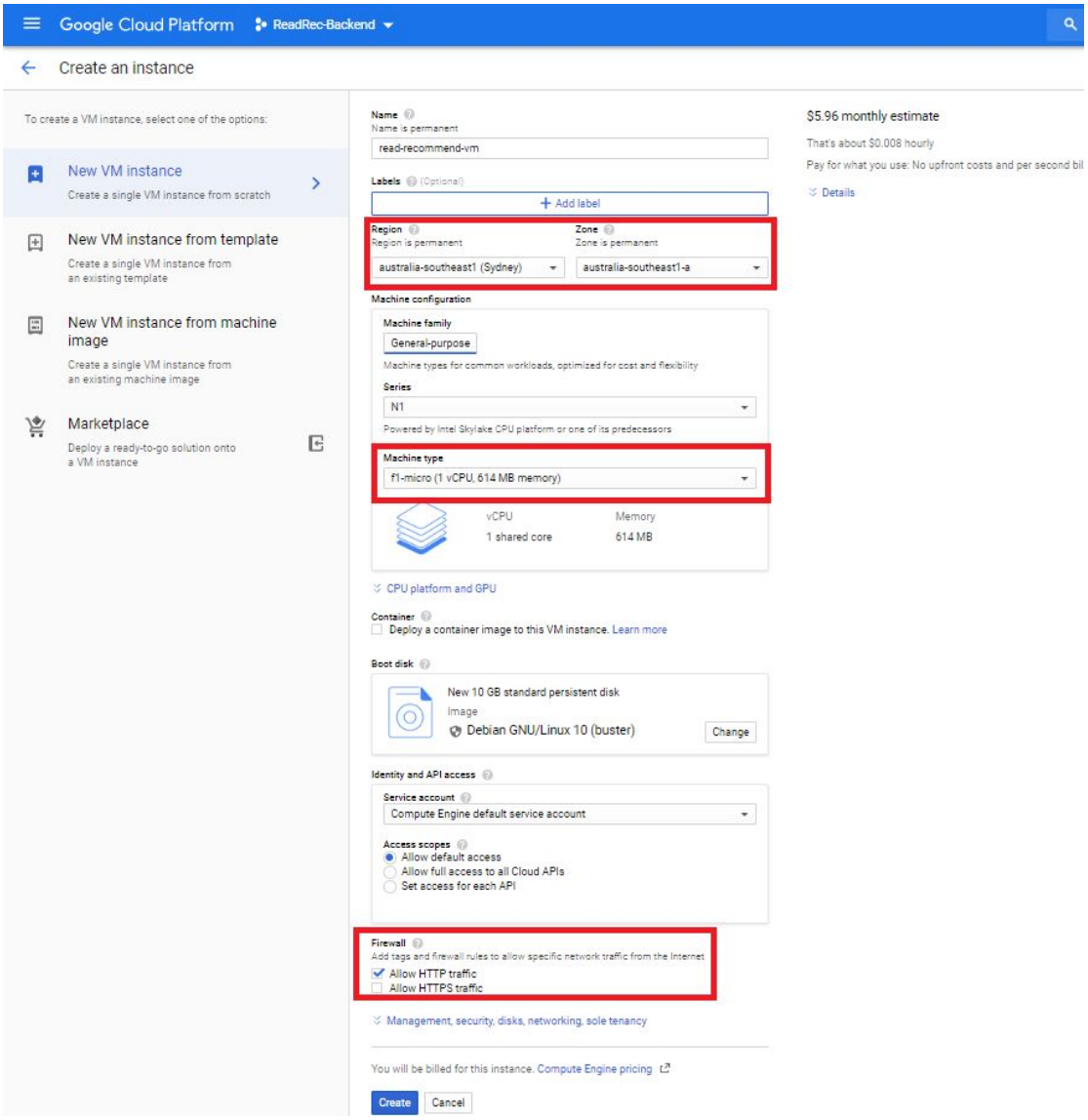
1. Go to Compute Engine -> VM instances



2. Click create instance button

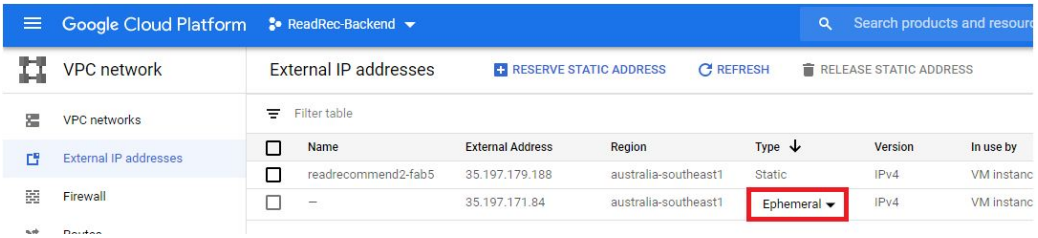


3. Choose Australia-southeast1 (Sydney) and Australia-southeast1-a which is the same as the database instance. Check Allow HTTP traffic box.



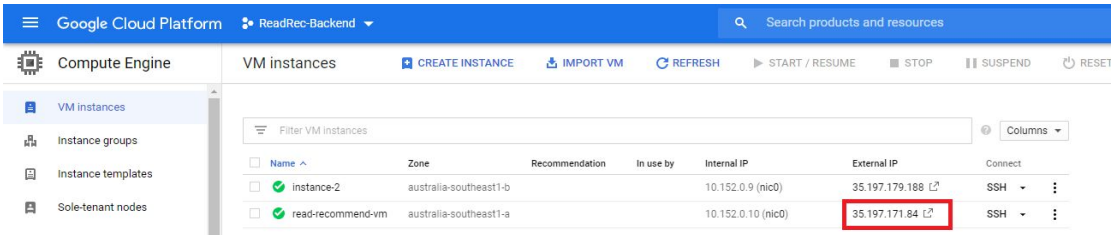
4. Click create button

5. Go to VPC network -> External IP addresses, change type to static

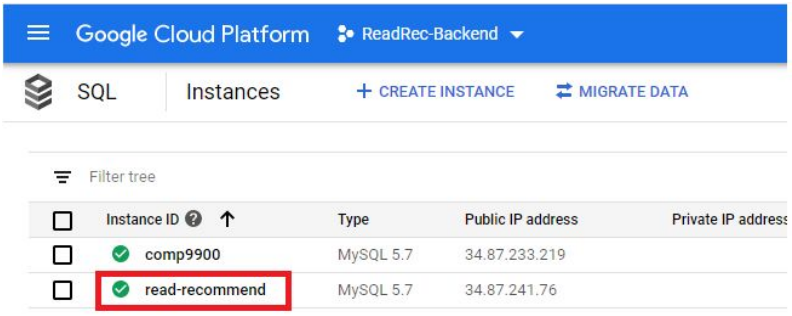


Cloud: Allow VM instance access the SQL database

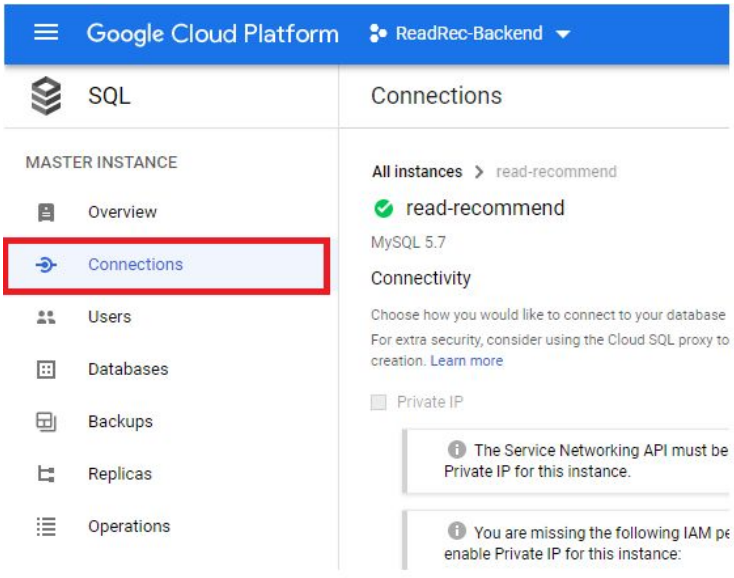
1. Go to the VM instances page, copy the external IP address.



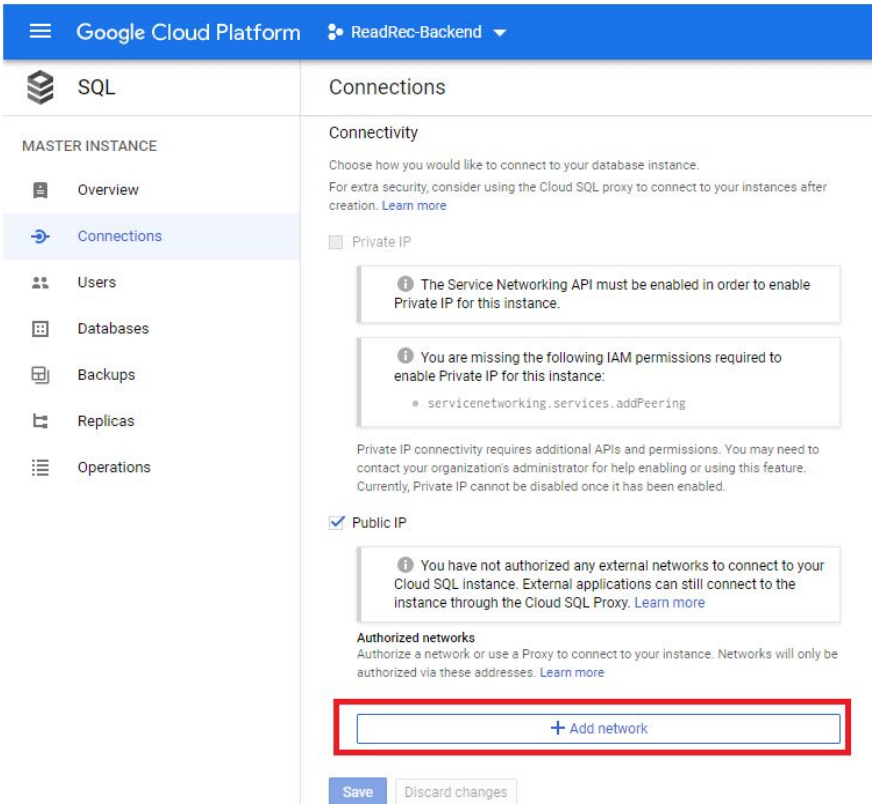
2. Go to SQL, click the database instance created before



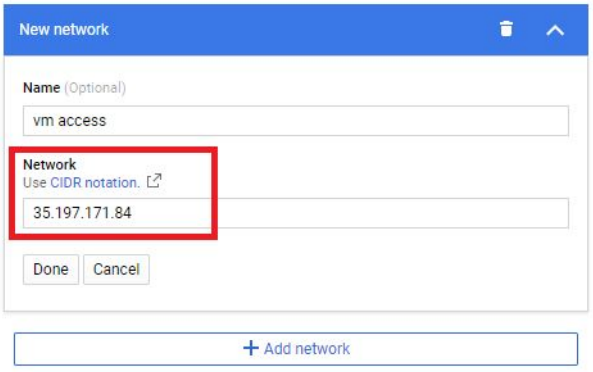
3. Go to Connections page



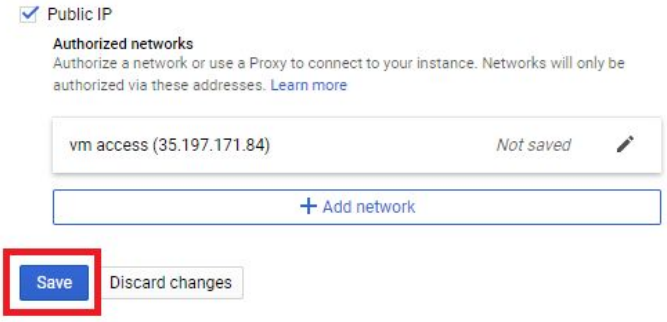
4. Click Add network button



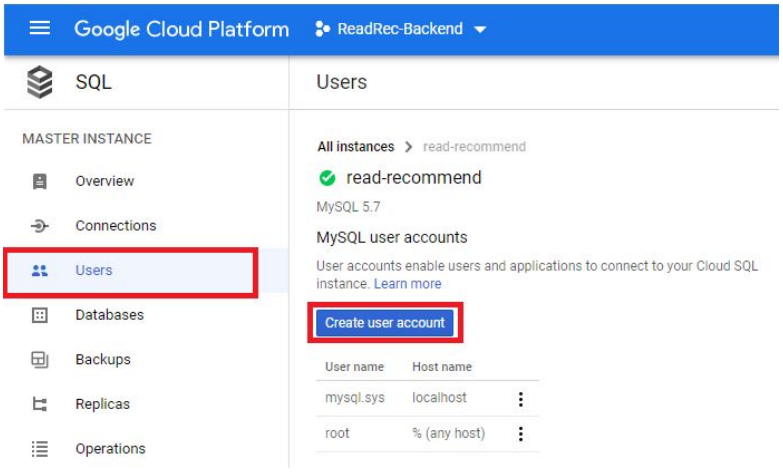
5. Copy the IP address and click done



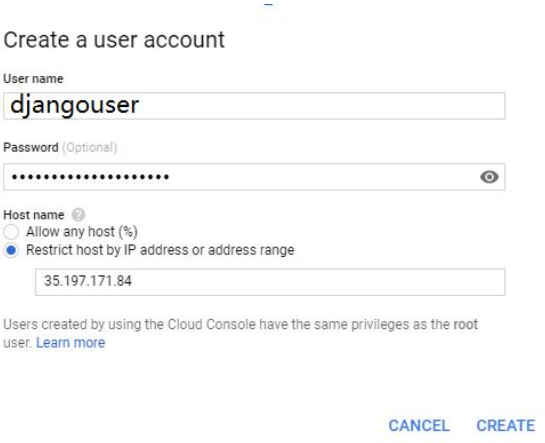
6. Click Save button



7. Go to users page, create a Django user

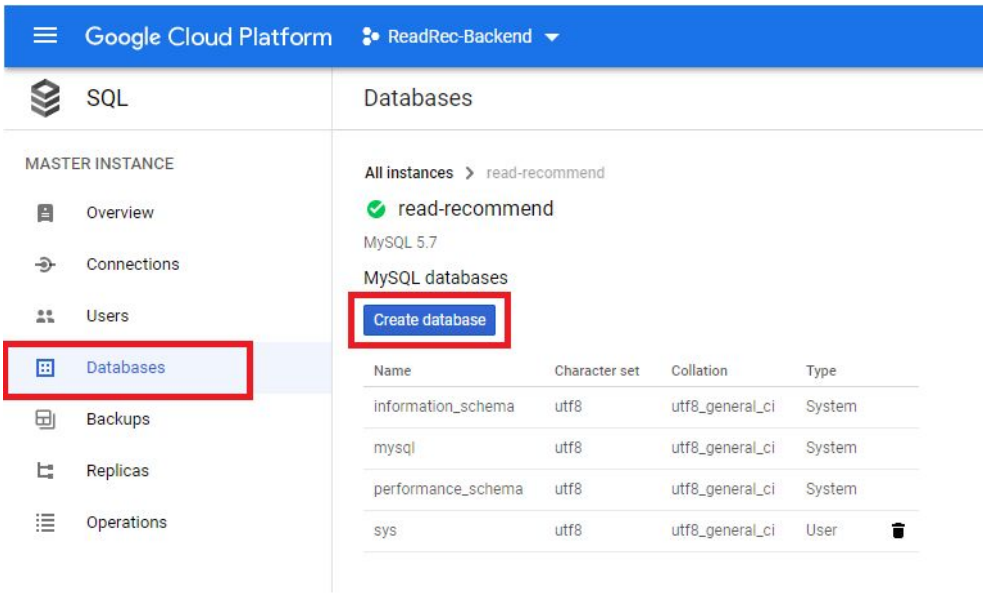


8. Enter the user name as djangouser, password as readrecommend_fab5, and copy IP address to restriction IP address. Click the create button.

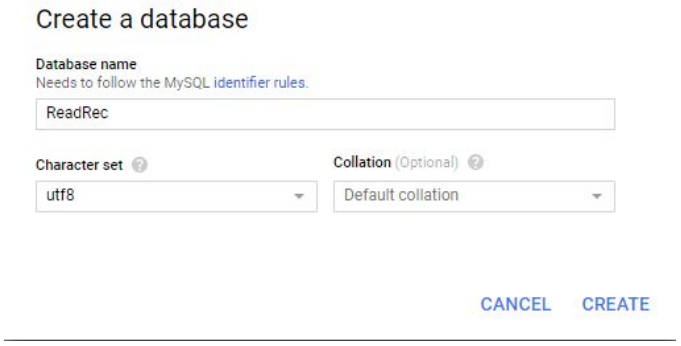


Cloud: Create a Google cloud database

1. Go to databases, click create database

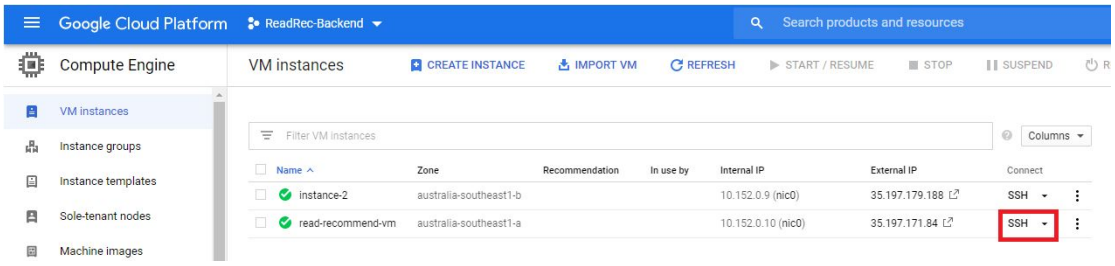


2. Enter the name as ReadRec, click create button

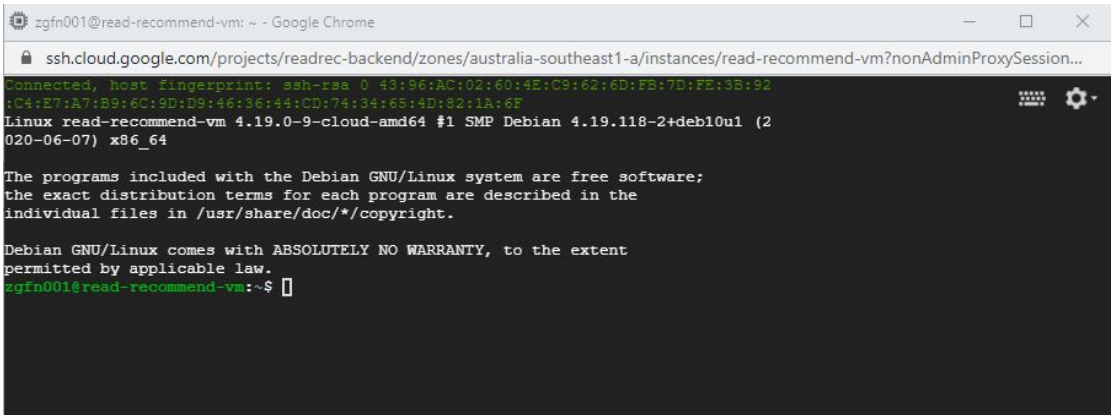


Cloud: Deploy vm instance to host Django application (easy)

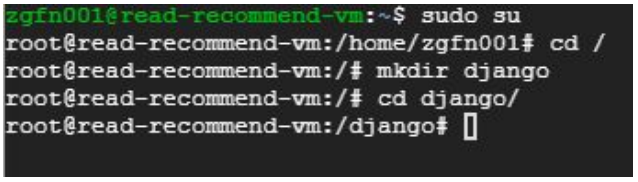
1. Go to VM instances, click ssh button



2. Now you can see a terminal window

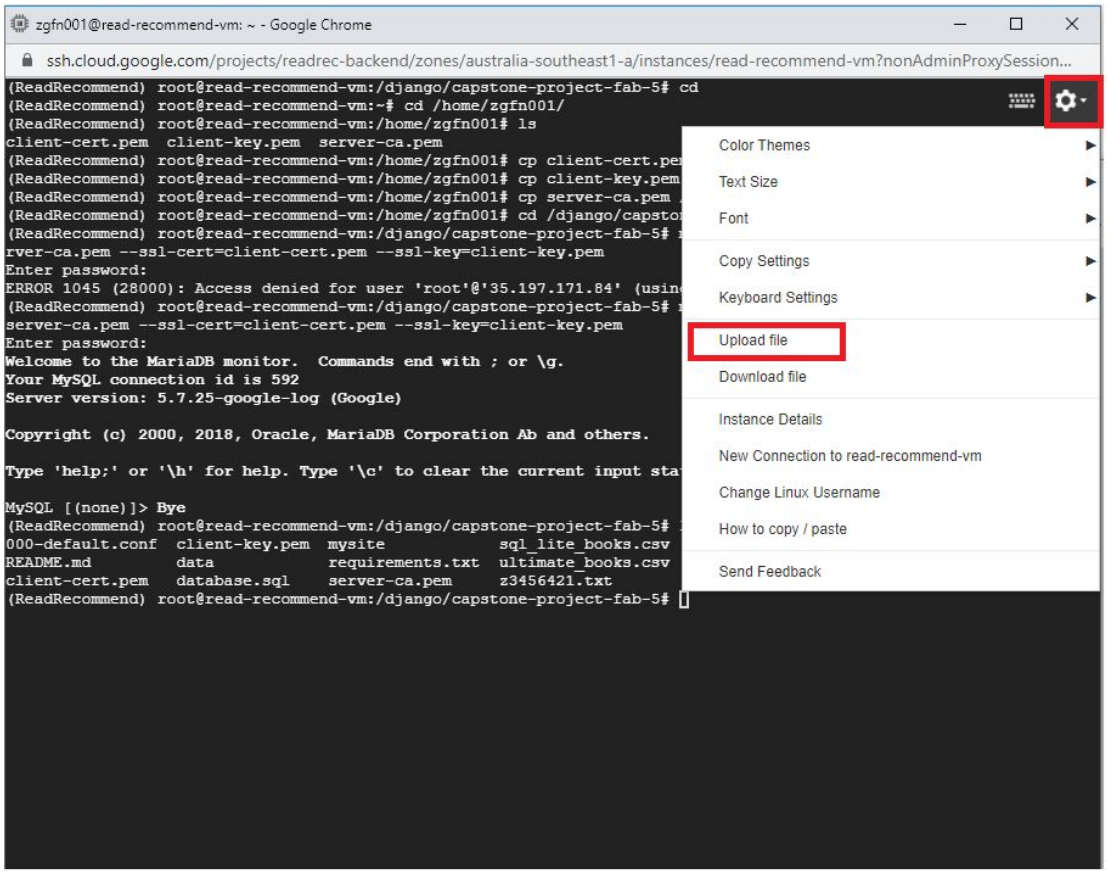


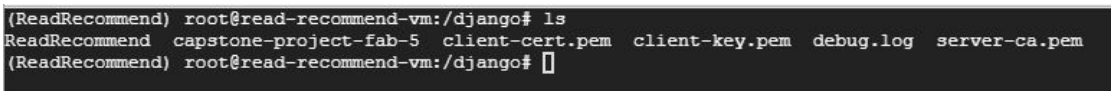
3. Use root user to create /django directory



4. change to /django directory

5. copy the three SQL SSL certificates into the /django directory. After uploading, they will be located in your home directory. Copy them into /django

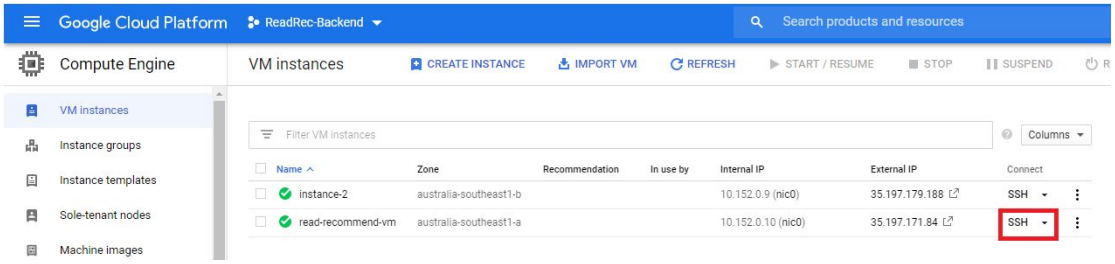




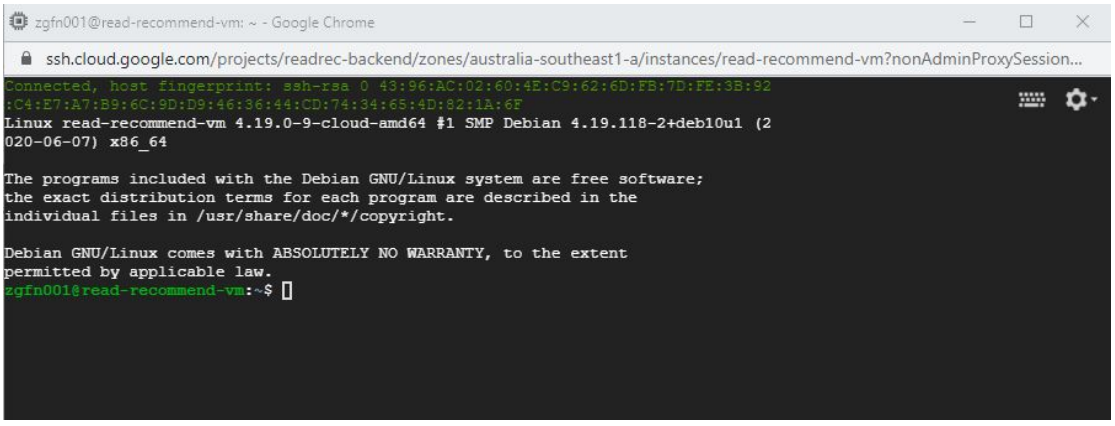
6. $ apt install git
7. $ git clone https://github.com/unsw-cse-capstone-project/capstone-project-fab-5.git
8. $ cd capstone-project-fab-5
9. $ bash setup.sh.
10. Setup.sh has created a virtual environment ReadRecommend for you. In future, if you want to modify the website, please source it then run python command.
11. check your website

Cloud: Deploy vm instance to host Django application (full steps)

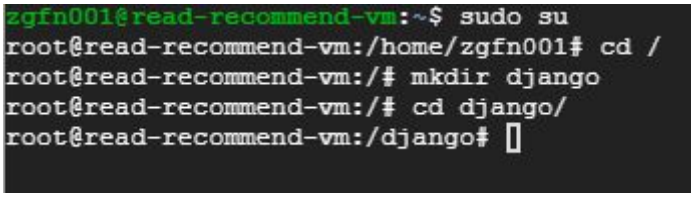
1. Go to VM instances, click ssh button



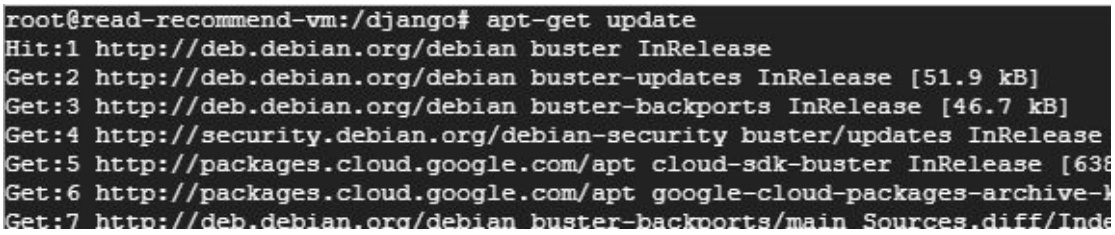
2. Now you can see a terminal window



3. Use root user to create /django directory



4. $ apt-get update



5. apt-get install python3-pip apache2 libapache2-mod-wsgi-py3

6. type y

```
root@read-recommend-vm:/django# apt-get install python3-pip apache2 libapache2-mod-wsgi-py3
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp
  cpp-8 dh-python dpkg-dev fakeroot g++ g++-8 gcc gcc-8 gir1.2-glib-2.0 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
  libasan5 libatomic1 libbinutils libbrotli1 libc-dev-bin libc6-dev libcc1-0 libdpkg-perl libexpat1-dev
  libfakeroot libfile-fcntllock-perl libgcc-8-dev libgdbm-compat4 libgirepository-1.0-1 libgomp1 libicu63
  libisl19 libitm1 libjansson4 liblocale-gettext-perl liblsan0 liblua5.2-0 libmpc3 libmpfr6 libmpx2 libperl5.28
  libpython3-dev libpython3.7 libpython3.7-dev libquadmath0 libstdc++-8-dev libtsan0 libubsan1 libxml2
  linux-libc-dev make manpages manpages-dev patch perl perl-modules-5.28 python-pip-whl python3-asn1crypto
  python3-cffi-backend python3-crypto python3-cryptography python3-dev python3-distutils python3-entrypoints
  python3-gi python3-keyring python3-keyrings.alt python3-lib2to3 python3-pkg-resources python3-secretstorage
  python3-setuptools python3-six python3-wheel python3-xdg python3.7-dev ssl-cert
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser binutils-doc cpp-doc gcc-8-locales
  debian-keyring g++-multilib g++-8-multilib gcc-8-doc libstdc++6-8-dbg gcc-multilib autoconf automake libtool
  flex bison gdb gcc-doc gcc-8-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan5-dbg
  liblsan0-dbg libtsan0-dbg libubsan1-dbg libmpx2-dbg libquadmath0-dbg glibc-doc git bzr libstdc++-8-doc make-doc
  ed diffutils-doc perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl libb-debug-perl
  liblocale-codes-perl python-crypto-doc python-cryptography-doc python3-cryptography-vectors gnome-keyring
  libkf5wallet-bin gir1.2-gnomekeyring-1.0 python-secretstorage-doc python-setuptools-doc openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils binutils binutils-common binutils-x86-64-linux-gnu
  build-essential cpp cpp-8 dh-python dpkg-dev fakeroot g++ g++-8 gcc gcc-8 gir1.2-glib-2.0
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libapache2-mod-wsgi-py3 libapr1
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libasan5 libatomic1 libbinutils libbrotli1 libc-dev-bin
  libc6-dev libcc1-0 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-8-dev libgdbm-compat4
  libgirepository-1.0-1 libgomp1 libicu63 libisl19 libitm1 libjansson4 liblocale-gettext-perl liblsan0
  liblua5.2-0 libmpc3 libmpfr6 libmpx2 libperl5.28 libpython3-dev libpython3.7 libpython3.7-dev libquadmath0
  libstdc++-8-dev libtsan0 libubsan1 libxml2 linux-libc-dev make manpages manpages-dev patch perl
  perl-modules-5.28 python-pip-whl python3-asn1crypto python3-cffi-backend python3-crypto python3-cryptography
  python3-dev python3-distutils python3-entrypoints python3-gi python3-keyring python3-keyrings.alt
  python3-lib2to3 python3-pip python3-pkg-resources python3-secretstorage python3-setuptools python3-six
  python3-wheel python3-xdg python3.7-dev ssl-cert
0 upgraded, 88 newly installed, 0 to remove and 1 not upgraded.
Need to get 124 MB of archives.
After this operation, 381 MB of additional disk space will be used.
Do you want to continue? [Y/n] 
```

7. install mysql client. $ apt install python3-dev libmariadb-dev

```
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# apt install python3-dev libmariadb-dev
Reading package lists... Done
Building dependency tree
Selecting previously unselected package mysql-common.
Preparing to unpack .../12-mysql-common_5.8+1.0.5_all.deb ...
Unpacking mysql-common (5.8+1.0.5) ...
Selecting previously unselected package mariadb-common.
Preparing to unpack .../13-mariadb-common_1%3a10.3.22-0+deb10u1_all.deb ...
Unpacking mariadb-common (1:10.3.22-0+deb10u1) ...
Selecting previously unselected package libmariadb3:amd64.
Preparing to unpack .../14-libmariadb3_1%3a10.3.22-0+deb10u1_amd64.deb ...
Unpacking libmariadb3:amd64 (1:10.3.22-0+deb10u1) ...
Selecting previously unselected package zlib1g-dev:amd64.
Preparing to unpack .../15-zlib1g-dev_1%3a1.2.11.dfsg-1_amd64.deb ...
Unpacking zlib1g-dev:amd64 (1:1.2.11.dfsg-1) ...
Selecting previously unselected package libmariadb-dev.
```

8. $ pip3 install virtualenv

```
root@read-recommend-vm:/django# pip3 install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/6f/a9/aec24edff88c204bb20b94ced8fd890176209aee20b4e701b796a6824e15/virtualenv-20.0.28-py2.py3-none-any.whl (4.9MB)
    100% |████████████████████████████████| 4.9MB 293kB/s
Collecting filelock<4,>=3.0.0 (from virtualenv)
  Downloading https://files.pythonhosted.org/packages/93/83/71a2ee6158bb9f39a90c0dea1637f81d5eef866e188e1971a1b1ab01a35a/filelock-3.0.12-py3-none-any.whl
Collecting appdirs<2,>=1.4.3 (from virtualenv)
  Downloading https://files.pythonhosted.org/packages/3b/00/2344469e2084fb287c2e0b57b72910309874c3245463acd6cf5e3db69324/appdirs-1.4.4-py2.py3-none-any.whl
Collecting importlib-metadata<2,>=0.12; python_version < "3.8" (from virtualenv)
  Downloading https://files.pythonhosted.org/packages/8e/58/cdea07eb51fc2b906db0968a94700866fc46249bdc75cac23f9d13168929/importlib_metadata-1.7.0-py2.py3-none-any.whl
Requirement already satisfied: six<2,>=1.9.0 in /usr/lib/python3/dist-packages (from virtualenv) (1.12.0)
Collecting distlib<1,>=0.3.1 (from virtualenv)
  Downloading https://files.pythonhosted.org/packages/f5/0a/490fa011d699bb5a5f3a0cf57de82237f52a6db9d40f33c53b2736c9a1f9/distlib-0.3.1-py2.py3-none-any.whl (335kB)
    100% |████████████████████████████████| 337kB 3.9MB/s
Collecting zipp>=0.5 (from importlib-metadata<2,>=0.12; python_version < "3.8"->virtualenv)
  Downloading https://files.pythonhosted.org/packages/b2/34/bfcb43cc0ba81f527bc4f40ef41ba2ff4080e047acb0586b56b3d017ace4/zipp-3.1.0-py3-none-any.whl
Installing collected packages: filelock, appdirs, zipp, importlib-metadata, distlib, virtualenv
Successfully installed appdirs-1.4.4 distlib-0.3.1 filelock-3.0.12 importlib-metadata-1.7.0 virtualenv-20.0.28 zipp-3.1.0
root@read-recommend-vm:/django# 
```

9. Inside django directory, create a virtual environment

```
root@read-recommend-vm:/django# virtualenv ReadRecommend
created virtual environment CPython3.7.3.final.0-64 in 157ms
  creator CPython3Posix(dest=/django/ReadRecommend, clear=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/root/.local/share/virtualenv)
    added seed packages: pip==20.1.1, setuptools==49.2.0, wheel==0.34.2
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
root@read-recommend-vm:/django#
```

10. Source the virtual environment

```
root@read-recommend-vm:/django# source ReadRecommend/bin/activate
(ReadRecommend) root@read-recommend-vm:/django#
```

11. Install git, type y

```
(ReadRecommend) root@read-recommend-vm:/django# apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man libcurl3-gnutls liberror-perl libpcre2-8-0
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man libcurl3-gnutls liberror-perl libpcre2-8-0
0 upgraded, 5 newly installed, 0 to remove and 1 not upgraded.
Need to get 7827 kB of archives.
After this operation, 39.3 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

12. Clone https://github.com/unsw-cse-capstone-project/capstone-project-fab-5.git

```
(ReadRecommend) root@read-recommend-vm:/django# git clone https://github.com/unsw-cse-capstone-project/capstone-project-fab-5.git
Cloning into 'capstone-project-fab-5'...
Username for 'https://github.com': z3456421
Password for 'https://z3456421@github.com':
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (79/79), done.
remote: Total 2894 (delta 65), reused 30 (delta 13), pack-reused 2799
Receiving objects: 100% (2894/2894), 7.33 MiB | 4.47 MiB/s, done.
Resolving deltas: 100% (1867/1867), done.
(ReadRecommend) root@read-recommend-vm:/django#
```

13. Add IP address of VM to ALLOWED_HOSTS

```
(ReadRecommend) root@read-recommend-vm:/django# cd capstone-project-fab-5/mysite/
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5/mysite# nano mysite/settings.py
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5/mysite#
```

```
ALLOWED_HOSTS = ['127.0.0.1', 'localhost', '35.197.171.84']
```

14. $ cd /django/capstone-project-fab-5

15. $ pip install –r requirements.txt

```
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# pip install -r requirements.txt
Collecting asgiref==3.2.10
  Using cached asgiref-3.2.10-py3-none-any.whl (19 kB)
Collecting Django==3.0.7
  Using cached Django-3.0.7-py3-none-any.whl (7.5 MB)
Collecting joblib==0.16.0
  Using cached joblib-0.16.0-py3-none-any.whl (300 kB)
Collecting mysqlclient==1.4.6
  Using cached mysqlclient-1.4.6.tar.gz (85 kB)
Collecting numpy==1.19.1
  Downloading numpy-1.19.1-cp37-cp37m-manylinux2010_x86_64.whl (14.5 MB)
     |████████████████████████████████| 14.5 MB 5.4 kB/s
Collecting Pillow==7.1.2
  Downloading Pillow-7.1.2-cp37-cp37m-manylinux1_x86_64.whl (2.1 MB)
     |████████████████████████████████| 2.1 MB 36.6 MB/s
Collecting pytz==2020.1
  Downloading pytz-2020.1-py2.py3-none-any.whl (510 kB)
     |████████████████████████████████| 510 kB 33.8 MB/s
Collecting scikit-learn==0.23.1
  Downloading scikit_learn-0.23.1-cp37-cp37m-manylinux1_x86_64.whl (6.8 MB)
     |████████████████████████████████| 6.8 MB 28.3 MB/s
Collecting scipy==1.5.2
  Downloading scipy-1.5.2-cp37-cp37m-manylinux1_x86_64.whl (25.9 MB)
     |████████████████████████████████| 25.9 MB 4.5 kB/s
Collecting sqlparse==0.3.1
  Downloading sqlparse-0.3.1-py2.py3-none-any.whl (40 kB)
     |████████████████████████████████| 40 kB 422 kB/s
Collecting threadpoolctl==2.1.0
  Downloading threadpoolctl-2.1.0-py3-none-any.whl (12 kB)
Building wheels for collected packages: mysqlclient
  Building wheel for mysqlclient (setup.py) ... done
  Created wheel for mysqlclient: filename=mysqlclient-1.4.6-cp37-cp37m-linux_x86_64.whl size=111639 sha256=85d09e46
4fed91c58f7250c8e90cd4b229213a65c1c988a35fcf36b36b8a8118
  Stored in directory: /root/.cache/pip/wheels/0d/77/50/313243794eb2e9f37ca5cbb677288009b2828c22f5f03dc7ff
Successfully built mysqlclient
Installing collected packages: asgiref, pytz, sqlparse, Django, joblib, mysqlclient, numpy, Pillow, scipy, threadpo
olctl, scikit-learn
Successfully installed Django-3.0.7 Pillow-7.1.2 asgiref-3.2.10 joblib-0.16.0 mysqlclient-1.4.6 numpy-1.19.1 pytz-2
020.1 scikit-learn-0.23.1 scipy-1.5.2 sqlparse-0.3.1 threadpoolctl-2.1.0
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5#
```

16. Install mysql-client (optional)

```
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# apt install default-mysql-client
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libaio1 libconfig-inifiles-perl libdbd-mysql-perl libdbi-perl libreadline5 libsnappy1v5 libterm-readkey-perl
  mariadb-client-10.3 mariadb-client-core-10.3
Suggested packages:
  libclone-perl libmldbm-perl libnet-daemon-perl libsql-statement-perl
The following NEW packages will be installed:
  default-mysql-client libaio1 libconfig-inifiles-perl libdbd-mysql-perl libdbi-perl libreadline5 libsnappy1v5
  libterm-readkey-perl mariadb-client-10.3 mariadb-client-core-10.3
0 upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 7990 kB of archives.
After this operation, 54.4 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

17. Test connection $ python manage.py dbshell

```
(ReadRecommend) root@read-recommend-vm:/django# cd /django/capstone-project-fab-5/mysite/
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5/mysite# python manage.py dbshell
Are you on localhost? (y/n): n
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 707
Server version: 5.7.25-google-log (Google)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [ReadRec]>
```

18. Copy 000-defualt.conf to /etc/apache2/sites-available

```
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# ls
000-default.conf  data          mysite           sql_lite_books.csv  z3456421.txt  z5199285.txt  z5222191.txt
README.md         database.sql  requirements.txt  ultimate_books.csv  z5180907.txt  z5219071.txt
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# cp 000-default.conf /etc/apache2/sites-
sites-available/ sites-enabled/
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# cp 000-default.conf /etc/apache2/sites-avail
able/
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# ls -l /etc/apache2/sites-available/
total 12
-rw-r--r-- 1 root root 2105 Jul 28 02:40 000-default.conf
-rw-r--r-- 1 root root 6338 Apr  2  2019 default-ssl.conf
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5#
```

19. Migrate models

```
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# cd /django/capstone-project-fab-5/mysite/
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5/mysite# python manage.py migrate
Are you on localhost? (y/n): n
Operations to perform:
  Apply all migrations: account, admin, auth, books, contenttypes, review, sessions, usercollections
Running migrations:
  Applying account.0001_initial... OK
  Applying contenttypes.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying books.0001_initial... OK
  Applying books.0002_auto_20200629_0150... OK
  Applying books.0003_auto_20200714_2324... OK
  Applying review.0001_initial... OK
  Applying review.0002_review_createddate... OK
  Applying sessions.0001_initial... OK
  Applying usercollections.0001_initial... OK
  Applying usercollections.0002_auto_20200701_0015... OK
  Applying usercollections.0003_auto_20200701_0059... OK
  Applying usercollections.0004_auto_20200701_0113... OK
  Applying usercollections.0005_booksadd... OK
  Applying usercollections.0006_auto_20200708_0601... OK
  Applying usercollections.0007_remove_collection_books... OK
  Applying usercollections.0008_collection_books... OK
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5/mysite#
```

20. Change permission of /django/debug.log

$ chown :www-data /django/debug.log

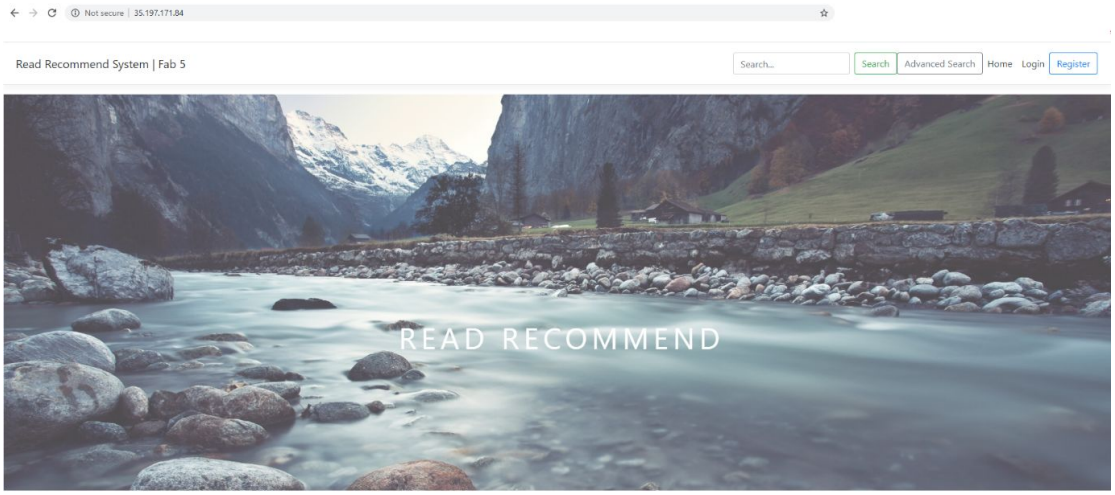
$ chmod g+w /django/debug.log

```
(ReadRecommend) root@read-recommend-vm:/django# ls -l
total 88
drwxr-xr-x 4 root root      4096 Jul 28 01:19 ReadRecommend
-rw-r--r-- 1 root root      5430 Jul 28 03:14 access.log
drwxr-xr-x 5 root root      4096 Jul 28 02:37 capstone-project-fab-5
-rw-r--r-- 1 root root      1223 Jul 28 02:20 client-cert.pem
-rw-r--r-- 1 root root      1674 Jul 28 02:20 client-key.pem
-rw-rw-r-- 1 root www-data 33593 Jul 28 03:13 debug.log
-rw-r--r-- 1 root root     24505 Jul 28 03:09 error.log
-rw-r--r-- 1 root root      1272 Jul 28 02:20 server-ca.pem
```

21. Restart the apache2 server

```
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5# service apache2 restart
(ReadRecommend) root@read-recommend-vm:/django/capstone-project-fab-5#
```

22. Go to browser to check the website



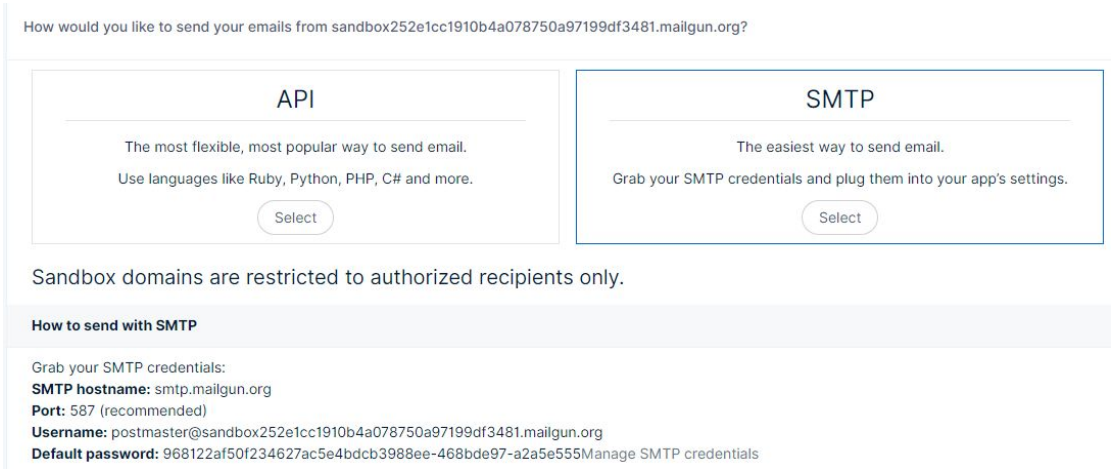
Cloud: Send Email

Use mailgun by applying the instruction from google mailgun documentation.

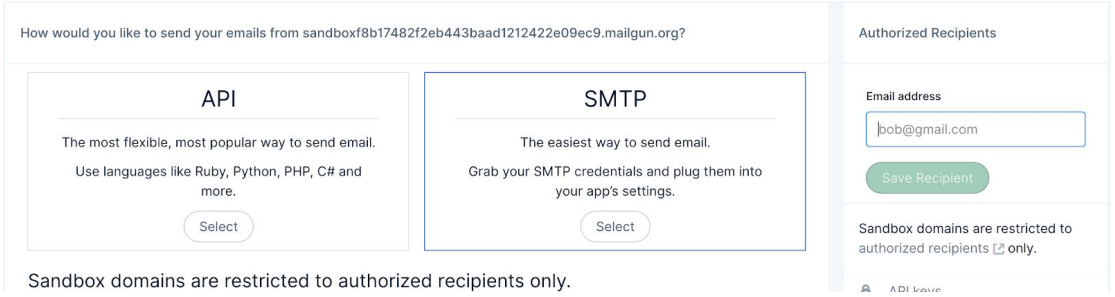
Source
https://cloud.google.com/compute/docs/tutorials/sending-mail/using-mailgun

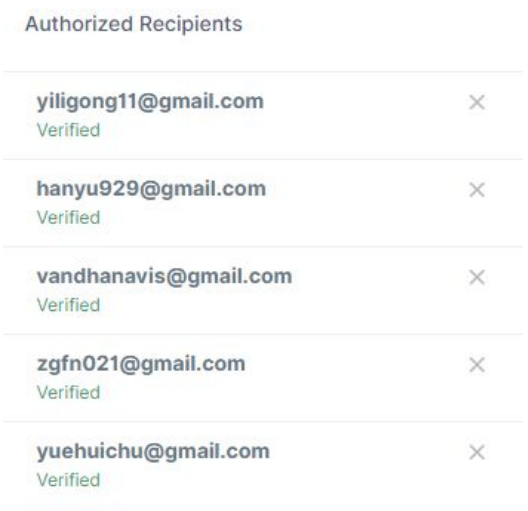
After creating a mailgun account, and installation, you need to change the settings.py in mysite directory. Copy the email host user and password from mailgun.

```
# Email
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.mailgun.org'
EMAIL_PORT = 2525
EMAIL_HOST_USER = 'postmaster@sandbox252e1cc1910b4a078750a97199df3481.mailgun.org'
EMAIL_HOST_PASSWORD = '968122af50f234627ac5e4bdcb3988ee-468bde97-a2a5e555'
DEFAULT_FROM_EMAIL = 'norply@readrecommend-fab5.com.au'
```



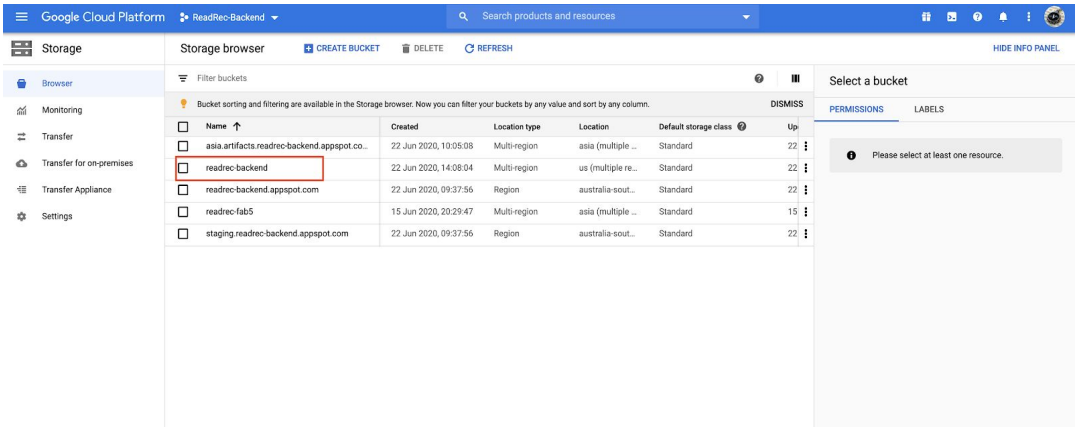
Since it is the testing version, when you want to test the sending email function, make sure to add authorized recipients like this.



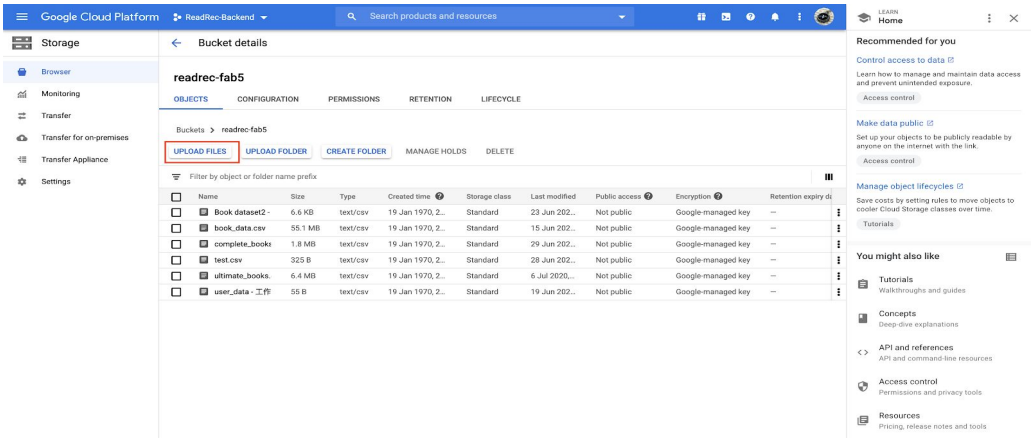

Authorized Recipients
yiligong11@gmail.com
Verified
hanyu929@gmail.com
Verified
vandhanavis@gmail.com
Verified
zgfn021@gmail.com
Verified
yuehuichu@gmail.com
Verified

Cloud: Insert book data

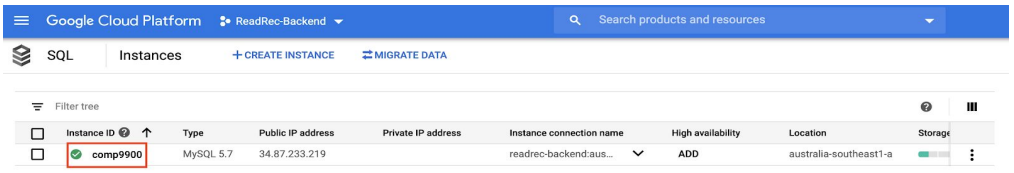
1. Upload the csv file you want to import into the database tables into cloud storage/bucket.



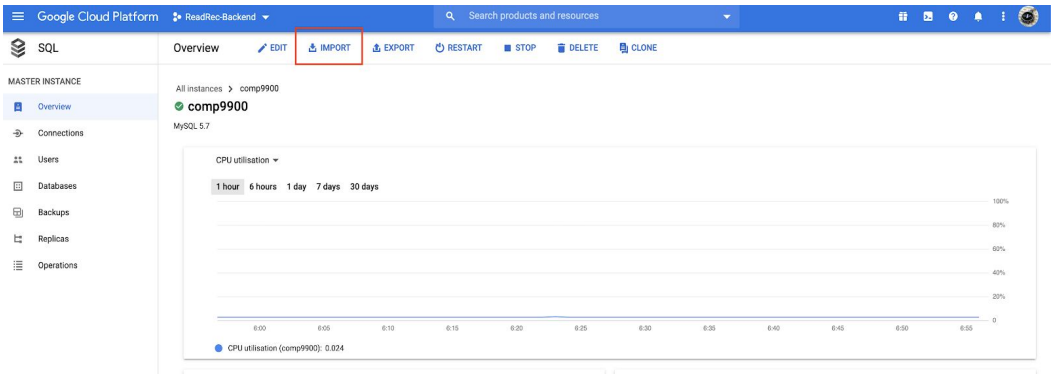
2. Insert file named “final_books.csv” from the data folder in the git repository into the preferred data bucket using upload files.



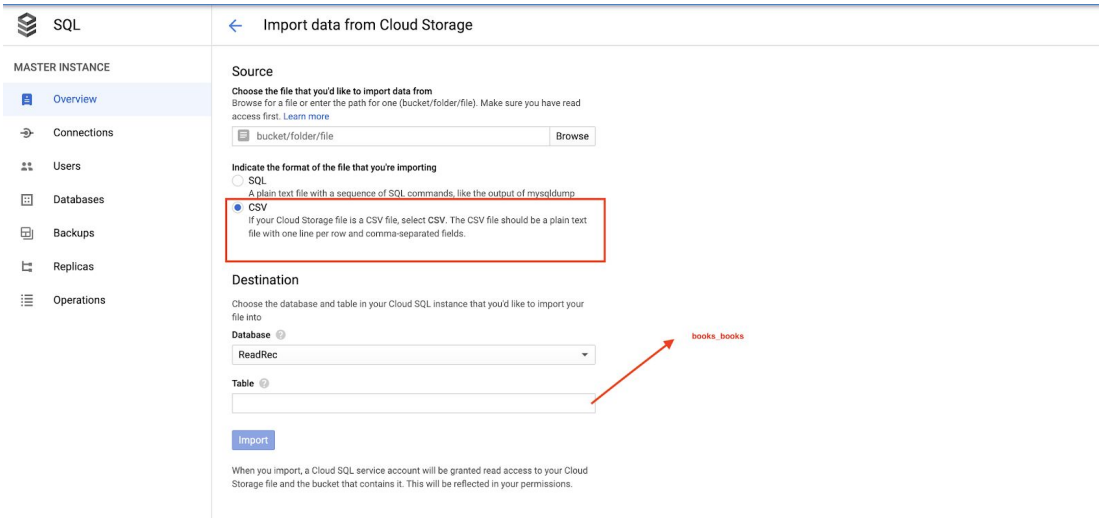
3. Switch tabs to go to the SQL Backend named “read-recommend”



4. Then click on the backend instance to go to this page.



5. Click on the import function to go to the next page. Choose the datafile named final_books.csv and type as CSV and not SQL. Type in the name of the table called “books_books”.



6. The data gets uploaded automatically after clicking import.

Conclusion and Future work

Thus the ReadRecommend system has been designed and developed to fit the requirements provided in the project description over a period of ten weeks. The system also offers novel features such as finding other similar users and books similar to what the user has read. The system also recommends books that are popular amongst other users and users can view how many people have read or want to read a given book. It has addressed the issues in other platforms like GoodReads and Riffle and is equally scalable.

The current system can be further improved by allowing the Machine Learning algorithm to calculate real time correlation values as new books are added. Furthermore, the user profile could have visual dashboards of their yearly reading summary and allow users to post about their recent reads on their walls, similar to facebook, allowing other users to like and comment on their posts.

References

George, N. (2018, Dec 7) Django's Structure – A Heretic's Eye View - The Django Book. Access date: Aug 4, 2020, website: https://djangobook.com/mdj2-django-structure/

jQuery (2020, May 4) Download jQuery | jQuery. Access Date : Aug 4, 2020 , website : https://jquery.com/download/

django. (2020). Model field reference. Retrieved from django: https://docs.djangoproject.com/en/3.0/ref/models/fields/#field-types

jQuery AJAX Methods. (n.d.). W3schools. Retrieved July 3, 2020, from https://www.w3schools.com/jquery/jquery_ref_ajax.asp