MANIPAL
ACADEMY *of* HIGHER EDUCATION
(Deemed to be University under Section 3 of the UGC Act, 1956)

INSPIRED BY LIFE

MAD LAB PROJECT REPORT

B.TECH CCE SIXTH SEMESTER

NextUP : A Project Management Tool

A PROJECT REPORT

submitted by

Name	**Registration Number**
Swadhin Routray	180953212
Aryaman Singh	180953260
Saksham Mehta	180953172

ACKNOWLEDGEMENT

At the very outset, I would like to give the first honors to the Almighty who gave me the wisdom and knowledge to complete this project. We also express our gratitude to **Mrs. Nisha Shetty and Mrs. Anuradha Rao** Faculty Advisor and Project Guide for providing us with adequate facilities, ways and means by which we were able to complete this project.

ABSTRACT

NextUp is a Project Management tool which will facilitate easier job scheduling, task completion, faster planning and help you keep track of deadlines. It aims to help you keep track of day to day tasks and also lets you add in extra work as and when necessary. With the right planning, one can ensure that your work is delivered on time and within budget. Using project management methods, one can map their project’s journey from the outset and know in advance where the deadlines and projected spend are going to fall, so you can more efficiently allocate your resources, helping you to avoid delays and project overspend. Working together can be hard. With more efficient project management processes, you can reduce the complexity of collaboration, increase transparency, and ensure accountability, even when you’re working across teams or departments.

CONTENTS

LIST OF ABBREVIATIONS

Title	Description
Project	A entity that constitutes of tasks
Task	An individual entity that outlines the work
Database	Collection of information in a structured form
userID	A self generated user identification number
projectID	A self generated project identification number
taskID	A self generated task identification number

INTRODUCTION

NextUP is a Project Management tool that helps track the progress of a project using a stage system. It aims to improve the productivity in a project and also implement a structure for project task segregation. We aim to help users complete their personal as well as group tasks using this application. Project management application software is designed to plan and document project tasks and activities, build schedules and timelines, solve project issues, manage risks and threats, assign budgets and control costs, establish collaboration and cooperation between project participants, assure and control quality, assemble project teams and organize human resources, and share information. It allows you to take your project through all the stages of the project life cycle, from project conceptualization and initiation through project execution, control and completion.

LANGUAGE DESCRIPTION

The Application is written in **Dart v2** using the **Flutter** App Development Framework. We used the Flutter-Dart framework as we all were comfortable with the language and have past experience for the same. We chose flutter as it is one of the most in-demand languages in the technical world.

Features of Flutter :-

Flutter framework offers the following features to developers –

- Modern and reactive framework.
- Uses Dart programming language and it is very easy to learn.
- Fast development.
- Beautiful and fluid user interfaces.
- Huge widget catalog.
- Runs the same UI for multiple platforms.
- High performance application.

Advantages of Flutter :-

Flutter comes with beautiful and customizable widgets for high performance and outstanding mobile application. It fulfills all the custom needs and requirements. Besides these, Flutter offers many more advantages as mentioned below –

- Dart has a large repository of software packages which lets you extend the capabilities of your application.
- Developers need to write just a single code base for both applications (both Android and iOS platforms). Flutter may be extended to other platforms as well in the future.
- Flutter needs less testing. Because of its single code base, it is sufficient if we write automated tests once for both the platforms.
- Flutter’s simplicity makes it a good candidate for fast development. Its customization capability and extendibility makes it even more powerful.
- With Flutter, developers have full control over the widgets and its layout.
- Flutter offers great developer tools, with amazing hot reload and hot restart features.

DATABASE/BACKEND DESCRIPTION

The **API** for the application is written in the **Node.js** framework. Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. We wanted to build a custom server-side for our application as it gives us full control over our features.

The **authentication** has been implemented using session storage in **Redis**. Redis is an in-memory data structure store, used as a distributed, in-memory key–value database. We have implemented it in order to safeguard sessions as well as to implement conditional rendering depending upon the user in session.

The **database** that has been used is **MongoDB**. MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

Interaction between the Application and the backend is outlined in the diagram in **Figure 1.**

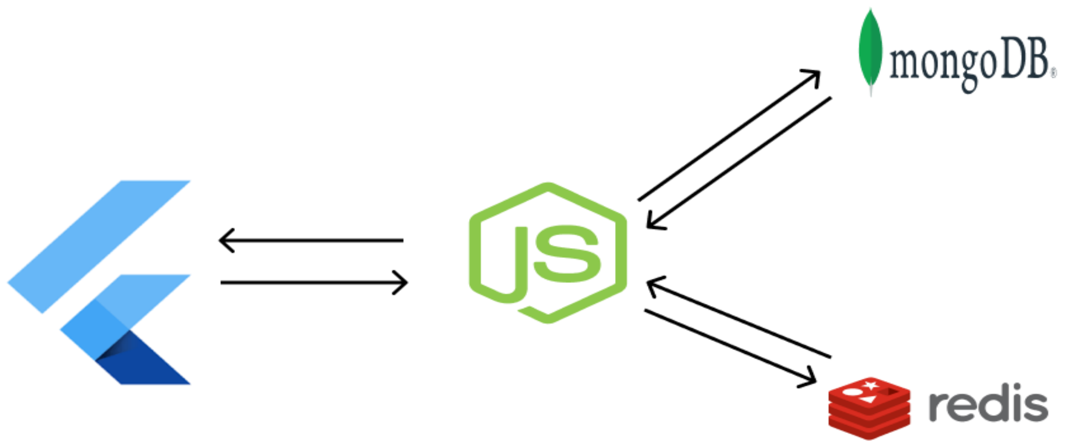


Figure 1.

PROJECT DESCRIPTION

Problem Statement

-A few problems our application attempts to overcome:

1. Creating a common workspace to track project's progress.
2. Ease of access for task completion.
3. Timely completion of group projects.
4. User Allocation tasks in a given project.
5. Grouping of tasks under the right stages for proper management.
6. Ease of addition of new employees/users to the project.

Objective

The primary objective of our project is the successful development of the user's project's procedures of initiation, planning, execution, regulation and closure as well as the guidance of the project team's operations towards achieving all the agreed upon goals within the set scope, time, quality and budget standards.

Proposed Methodology

Product Functions

Project Creation: The user, on logging in can create a new project. In order to add a new project, you require a project title and the deadline to the given project.

Task Addition: Once a new project is created, the user can add a new task to that particular project. To add a task, the user simply has to add a task title and task description. The task on creation, by default, goes into the ***Ideation*** stage of the project.

Task Deletion: The user, if he wishes to, can delete a particular task. This may be due to either typos in the title or description or just a redundant task deletion.

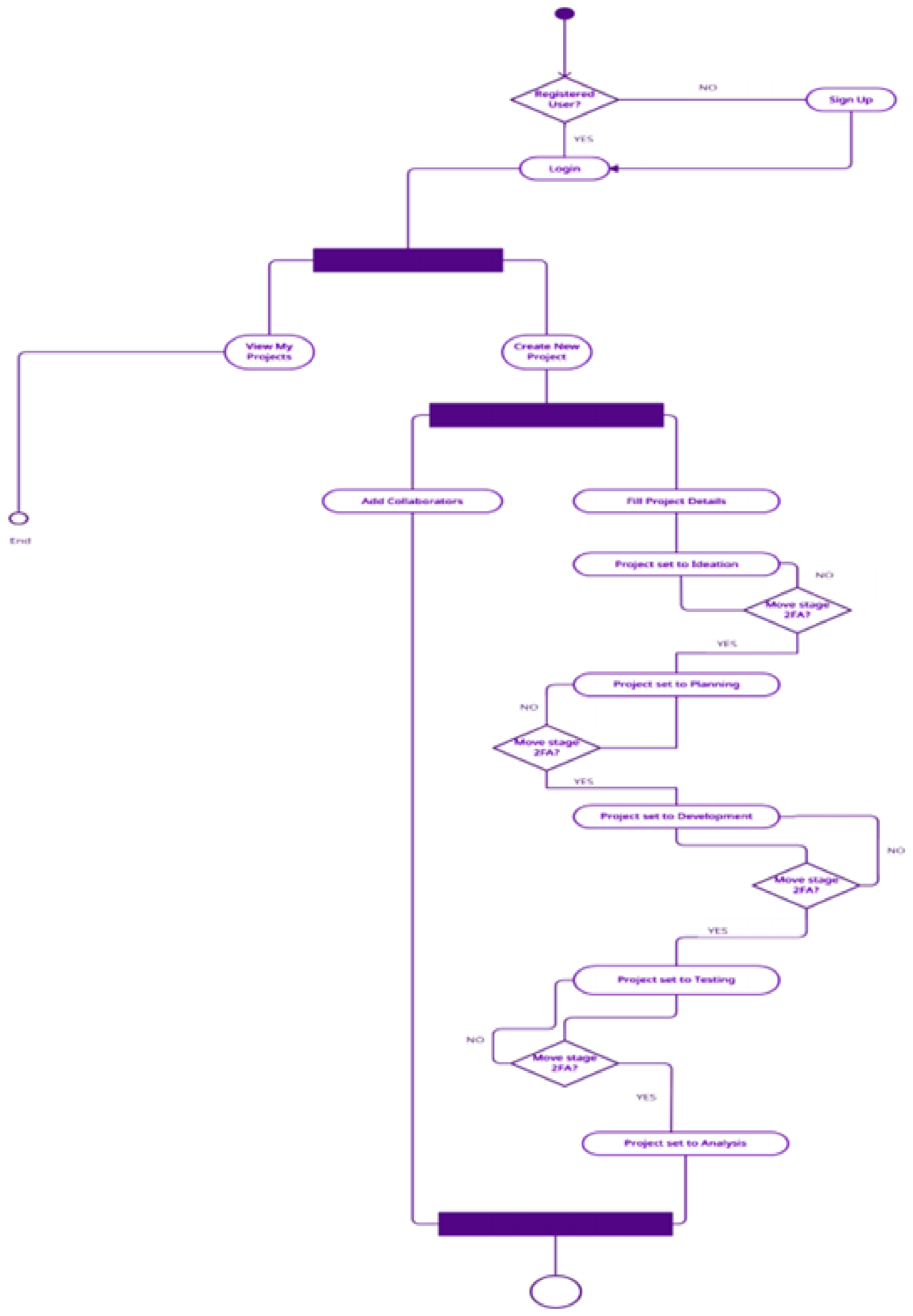
Move Task Stage: On completion of a task, the user can move the task to the next stage. Each task has only forward propagation, i.e, it can move from ***Ideation*** to ***Planning*** stage, but not the other way round for any stage. Hence, in order to remove any confusions and to properly verify task completion, it is safeguarded by a 2FA(2 Factor Authentication) method. It is implemented using an OTP, sent to the user via email, that has initiated the stage movement. This prevents any premature movement of tasks between stages. The ***Analysis*** stage is the last stage in any project.

Addition of Collaborator: The user can add a collaborator to a project. The prerequisite for this is that the user being added, should already exist in the database. Once added to the project, the collaborator can view all the tasks in the project and also he also has the ability to create new tasks, delete existing ones as well as move the tasks from one stage to another.

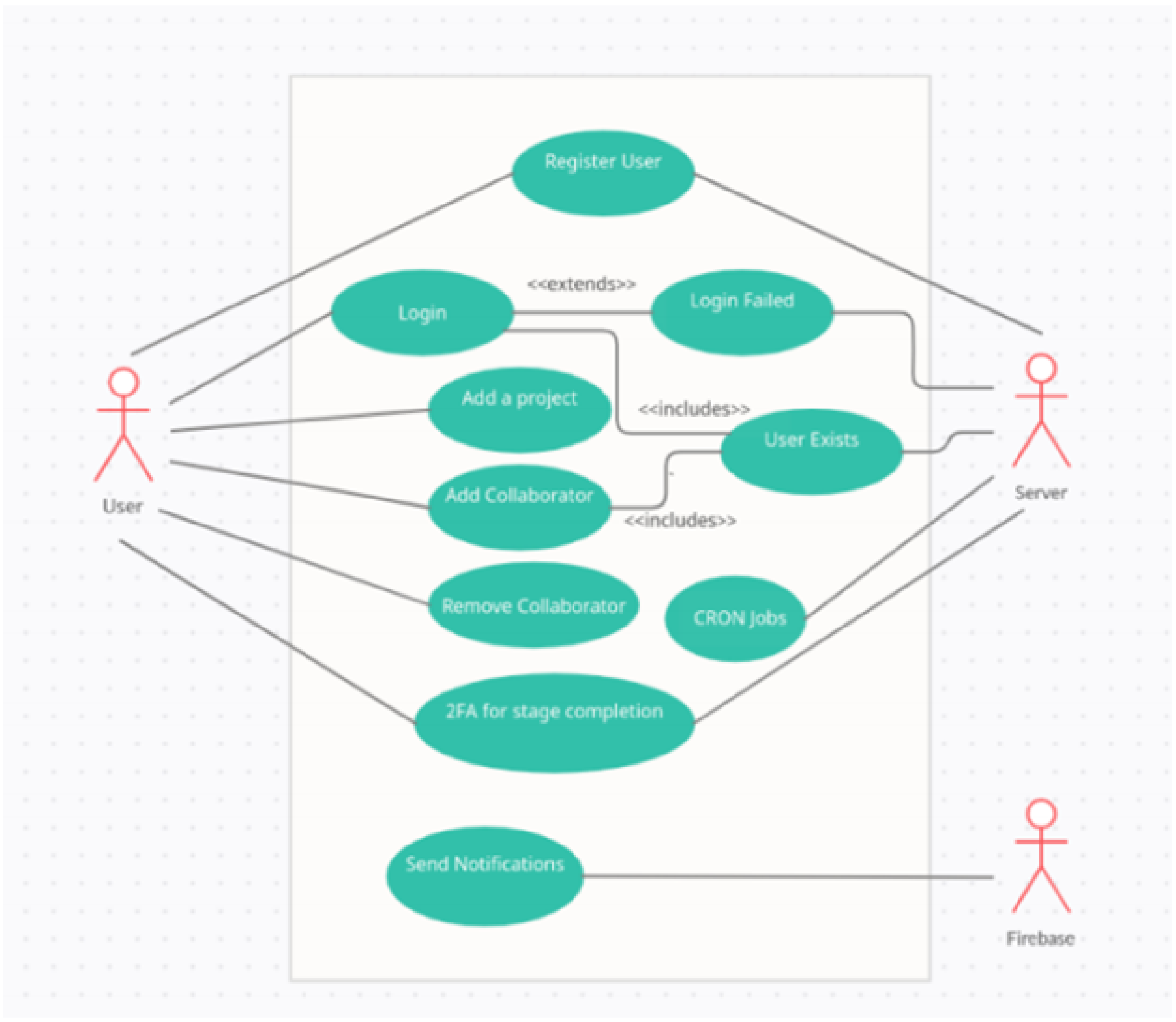
Removal of Collaborator: A collaborator can remove another collaborator from a project. Once removed from the project, the removed collaborator cannot access the project anymore.

DESIGN

1. Activity Diagram:



2. <u>UseCaseDiagram:</u>



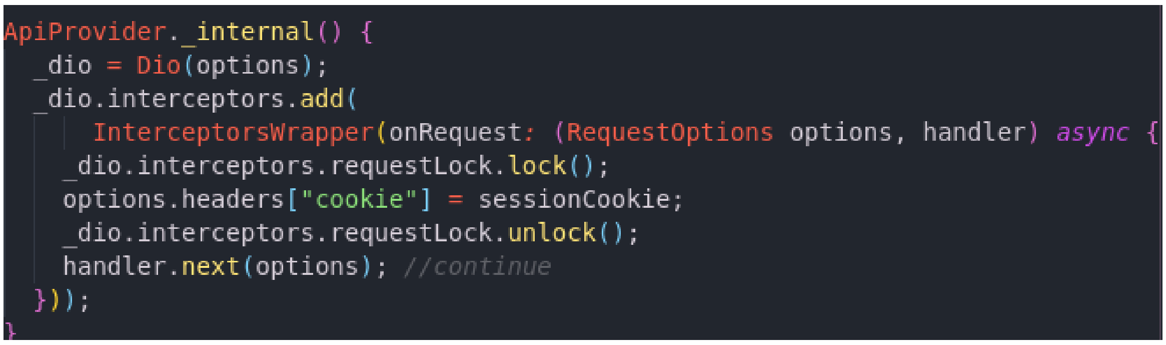
IMPLEMENTATION

In this section, we outline the implementations of the **Application**, the **API** implementation and the **Database** models and key queries.

Application Implementation:

This section outlines key Application implementations

- The application is implemented using the Flutter framework, using the Dart language.
- The authentication screens,i.e, the home screen, login screen and the sign up screens are implemented using basic TextField and FlatButtons.
- On login, we store the session cookie retrieved from the API(generated automatically by Redis) in the local storage of the application. This is achieved using the **shared-preferences** package. This is essential as it helps us auto login the user if he had previously signed in and hadn't logged out before closing the application.
- The API calls are made using the **dio** package. Dio is a powerful HTTP client for Dart, which supports Interceptors, Global Configuration, FormData, Request Cancellation, File Downloading, Timeout, etc. We use the interceptor feature to auto attach the session token in the header of the request to signify to the API that the user is logged in and that the sessionID is his unique identification ID. The implementation is shown below in **Figure 2.**

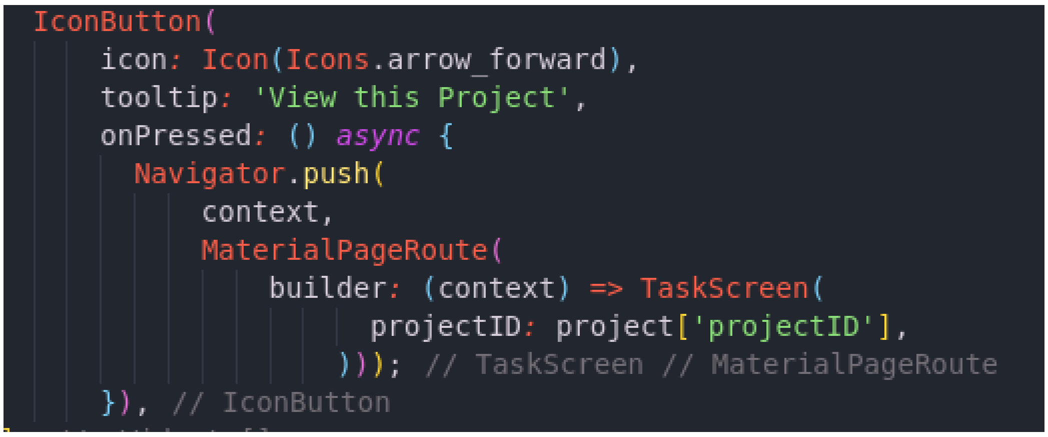


```
ApiProvider._internal() {
  _dio = Dio(options);
  _dio.interceptors.add(
      InterceptorsWrapper(onRequest: (RequestOptions options, handler) async {
    _dio.interceptors.requestLock.lock();
    options.headers["cookie"] = sessionCookie;
    _dio.interceptors.requestLock.unlock();
    handler.next(options); //continue
  }));
}
```

Figure 2.

- The Project Screen is implemented by fetching the session based user as stored in the local storage, and then displayed using the Future builder. The Collaborator and task screens are implemented in a similar way.

- The OTP initiation and completion is taken care of using custom API calls made by the application on move stage button click.
- Page Navigation is taken care of by the Navigator and Material functions that are inbuilt in Flutter. An implementation example is shown below in **Figure 3.**



```
IconButton(
    icon: Icon(Icons.arrow_forward),
    tooltip: 'View this Project',
    onPressed: () async {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => TaskScreen(
            projectID: project['projectID'],
        ))); // TaskScreen // MaterialPageRoute
    }), // IconButton
```

Figure 3

- Toasts are implemented in the application using the **fluttertoast** package. THis is shown when the user OTP is wrong or the user has logged in using invalid credentials.
- The application was built and run on an emulator to show the implementation.

API Implementation:

This section outlines key API implementations

- The API is implemented using Node.js while using the **Express.js** application framework. It is a back end web application framework for Node.js.
- We use the **mongoose** package as the ORM to communicate with the MongoDB database. It includes built-in validation of data and hels skip the request validation step.
- Authentication features have been implemented using the **redis**, **express-session** and **cookie-parser** packages. On saving the user session, we create an entry in the redis DB that persists until:

  The time to live (TTL) expires.

  OR

  The user logs out and the session is destroyed.
- On adding collaborators, we implement an updateOne() function on both the project document as well as the user document.
- Stage completion and Stage movement features are implemented using the OTP functionality.
- OTPs are sent to the users using a custom mailing system that has been implemented in the API. The implementation uses **Amazon Web Services' Simple Emailing Services** or simply **AWS SES**. The mailing functionality has been implemented for two major functions,i.e., addition of collaborator and OTP verification.
- Fetch routes have been implemented as well. These include fetching session based user projects, collaborators of a project and fetching tasks according to build stage to name a few.
- Some of these features utilize complex aggregation pipelines in order to fetch the data in the right format for ease of access on the application.

<u>Database Implementation:</u>

This section outlines key Database models and queries.

Our Application consists of three major models as outlined in the ER Diagram in **Figure 4.**

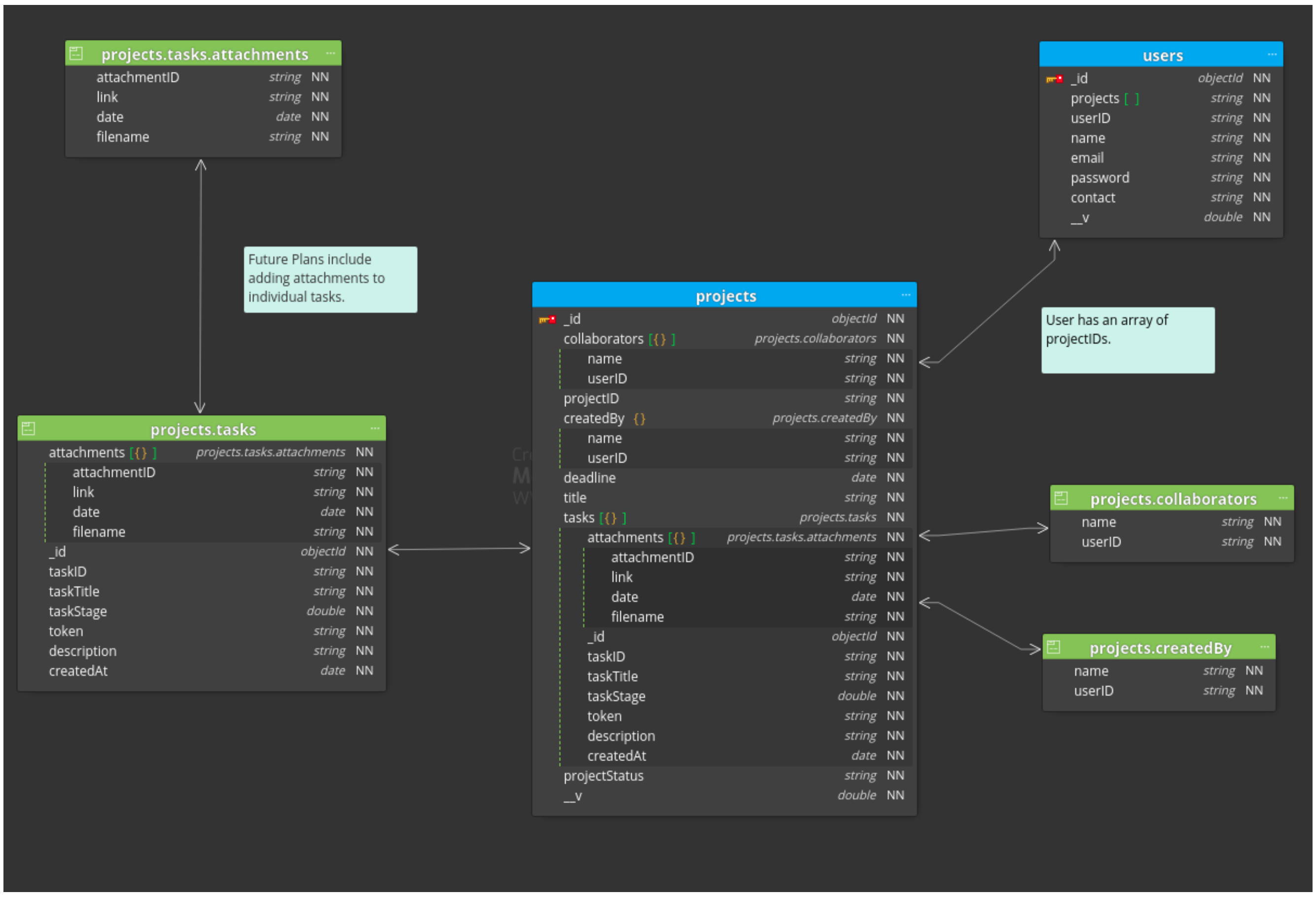


Figure 4

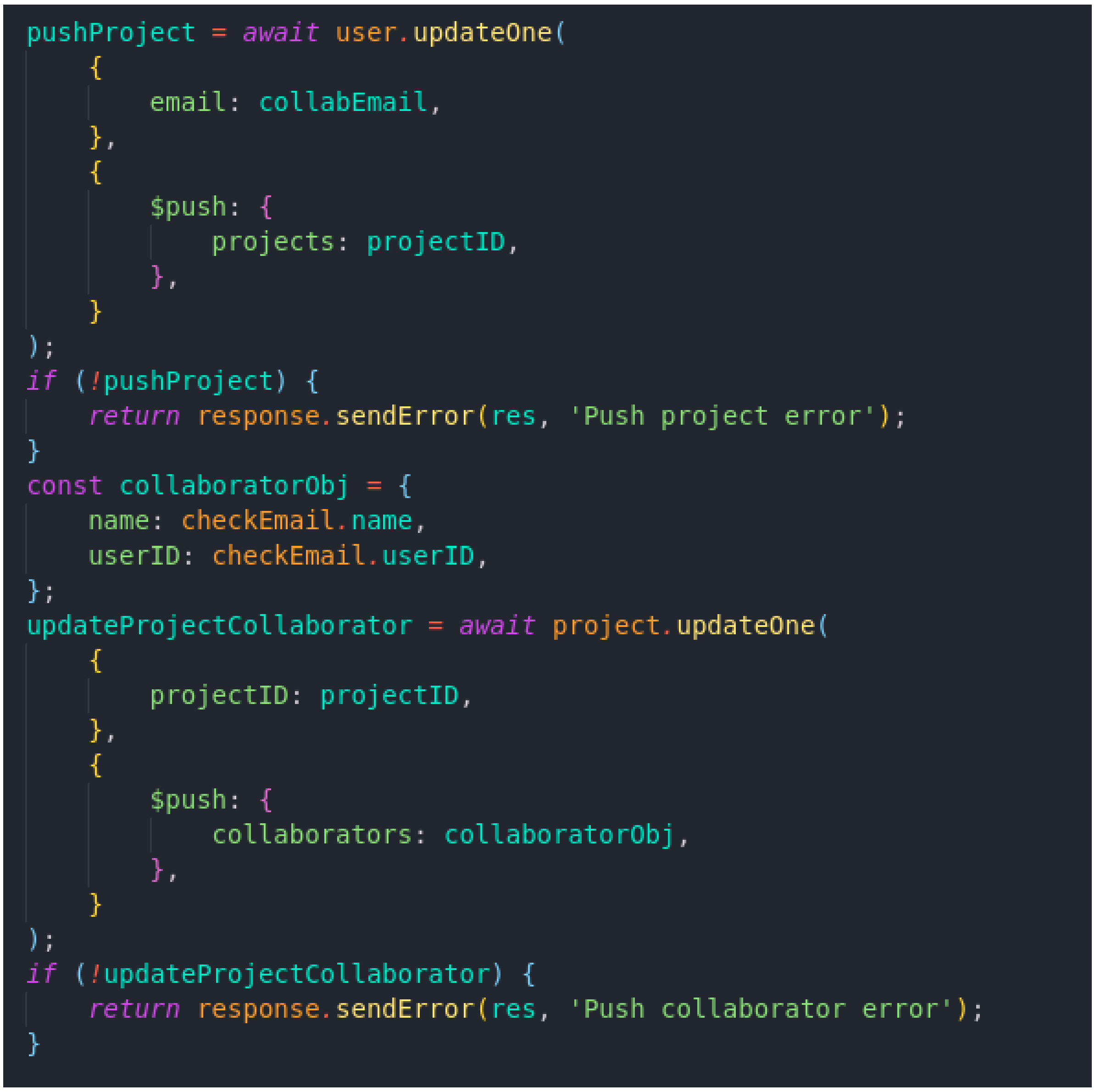
User Model: Includes the email, password, userID, contact and other details.

Project Model: Includes the project title, projectID, an array of the tasks model, collaborators array, deadline and creator credentials.

Task Model: Stored individual task titles and description along with otp token, which is refreshed on each stage move as well as on wrong otp code. It also includes an attachments array, which is in the works for future plans.

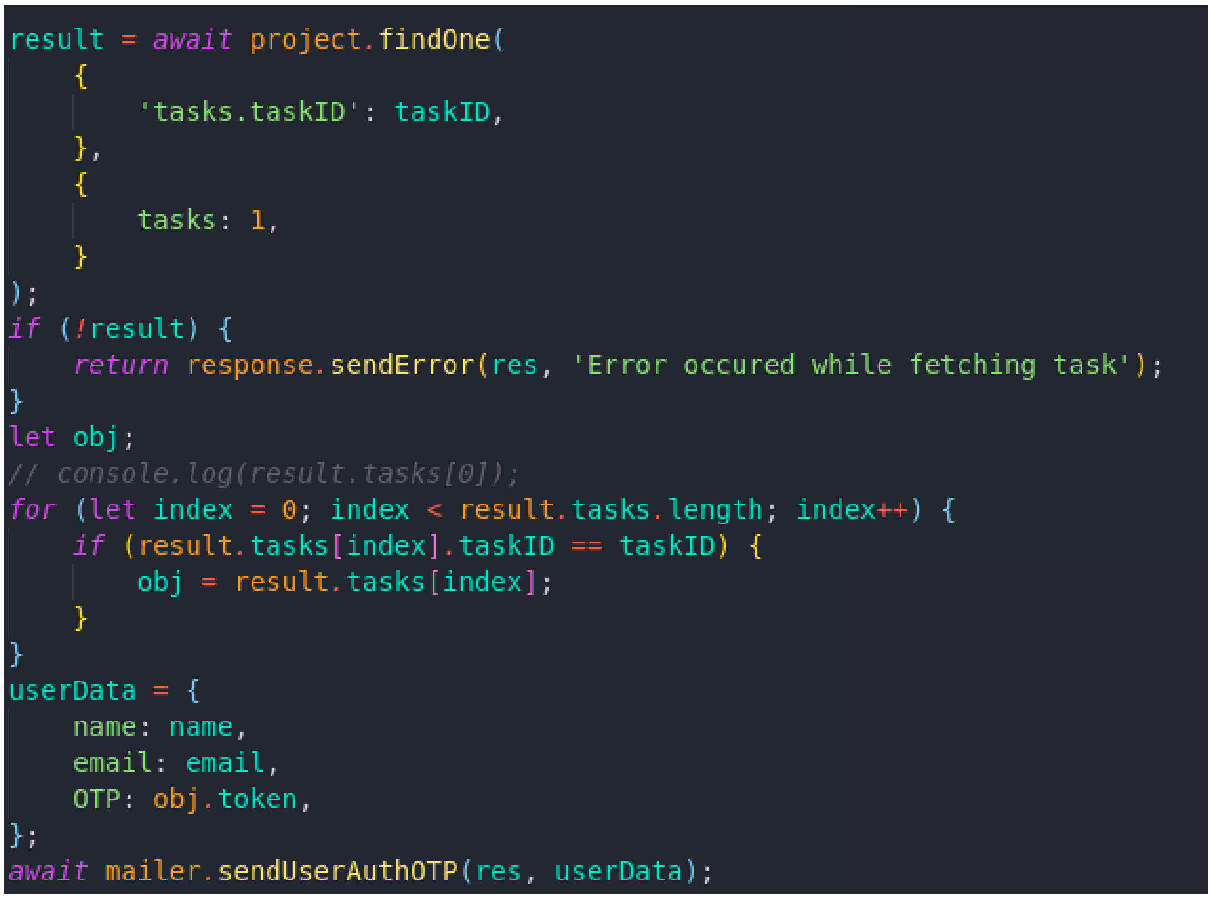
Important DB Queries to understand implementation depth:

- Collaborator updation in both user and project documents:



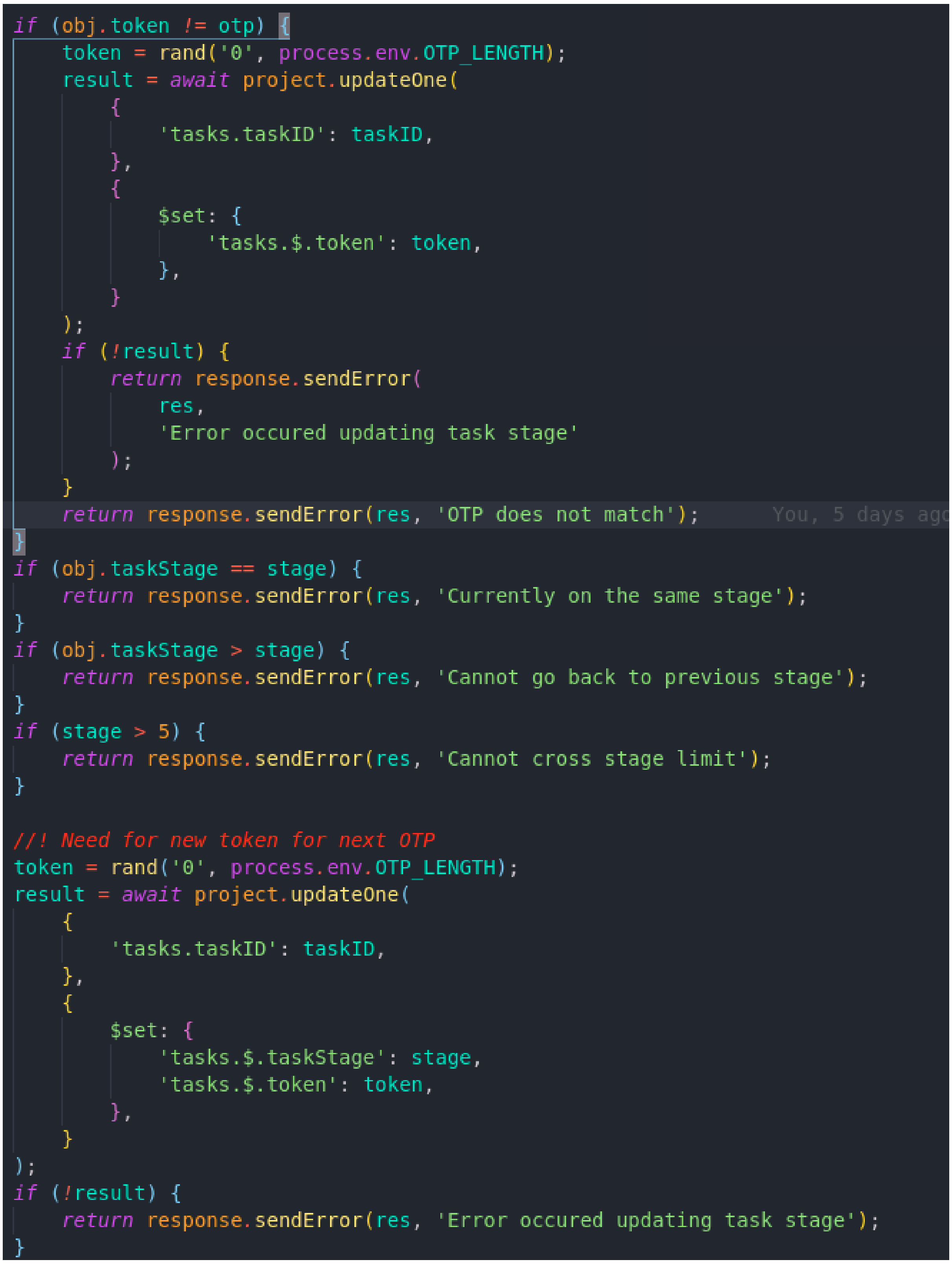
```
pushProject = await user.updateOne(
    {
        email: collabEmail,
    },
    {
        $push: {
            projects: projectID,
        },
    }
);
if (!pushProject) {
    return response.sendError(res, 'Push project error');
}
const collaboratorObj = {
    name: checkEmail.name,
    userID: checkEmail.userID,
};
updateProjectCollaborator = await project.updateOne(
    {
        projectID: projectID,
    },
    {
        $push: {
            collaborators: collaboratorObj,
        },
    }
);
if (!updateProjectCollaborator) {
    return response.sendError(res, 'Push collaborator error');
}
```

- Stage Initiation and OTP fetching:



```
result = await project.findOne(
    {
        'tasks.taskID': taskID,
    },
    {
        tasks: 1,
    }
);
if (!result) {
    return response.sendError(res, 'Error occured while fetching task');
}
let obj;
// console.log(result.tasks[0]);
for (let index = 0; index < result.tasks.length; index++) {
    if (result.tasks[index].taskID == taskID) {
        obj = result.tasks[index];
    }
}
userData = {
    name: name,
    email: email,
    OTP: obj.token,
};
await mailer.sendUserAuthOTP(res, userData);
```

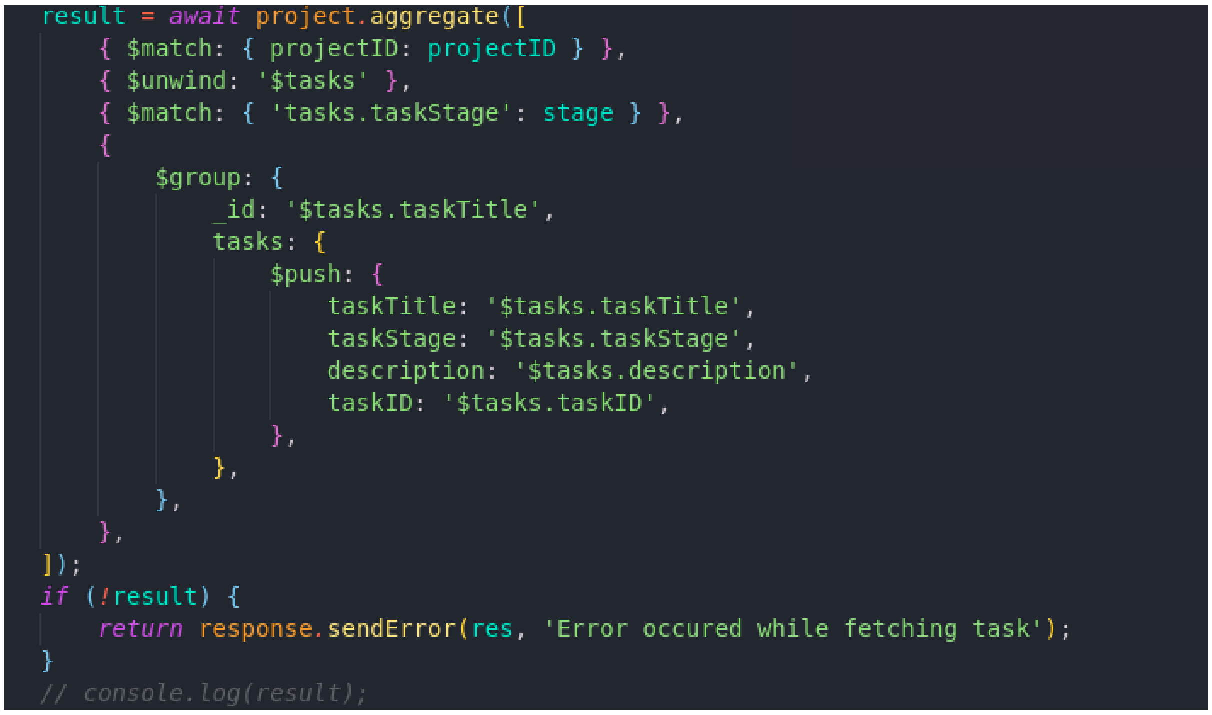
- Stage Completion and task stage updation as well as wrong OTP check and refresh OTP token:



```
if (obj.token != otp) {
    token = rand('0', process.env.OTP_LENGTH);
    result = await project.updateOne(
        {
            'tasks.taskID': taskID,
        },
        {
            $set: {
                'tasks.$.token': token,
            },
        }
    );
    if (!result) {
        return response.sendError(
            res,
            'Error occured updating task stage'
        );
    }
    return response.sendError(res, 'OTP does not match');      You, 5 days ago
}
if (obj.taskStage == stage) {
    return response.sendError(res, 'Currently on the same stage');
}
if (obj.taskStage > stage) {
    return response.sendError(res, 'Cannot go back to previous stage');
}
if (stage > 5) {
    return response.sendError(res, 'Cannot cross stage limit');
}

//! Need for new token for next OTP
token = rand('0', process.env.OTP_LENGTH);
result = await project.updateOne(
    {
        'tasks.taskID': taskID,
    },
    {
        $set: {
            'tasks.$.taskStage': stage,
            'tasks.$.token': token,
        },
    }
);
if (!result) {
    return response.sendError(res, 'Error occured updating task stage');
}
```

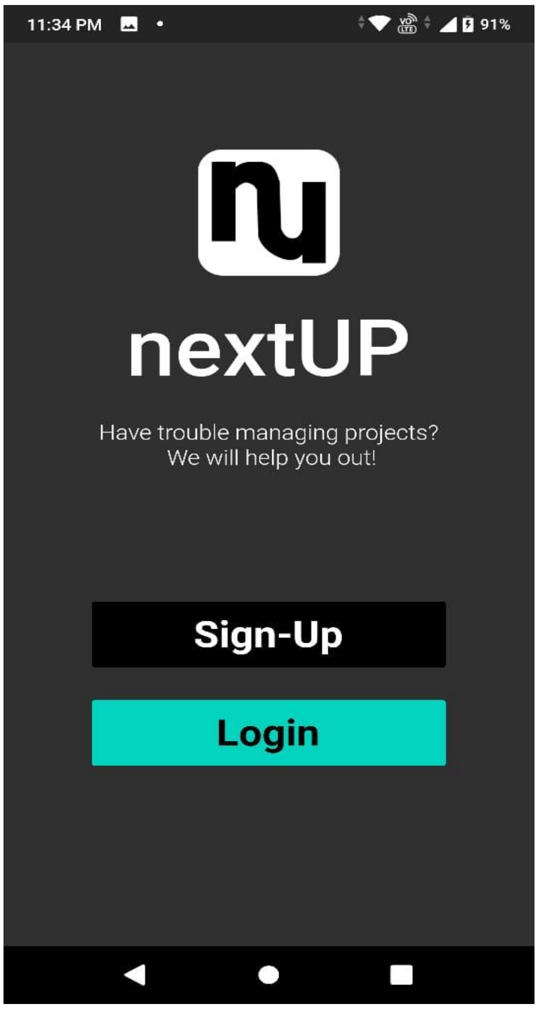
- Aggregation pipeline to fetch data from DB as required for easy display on the application:

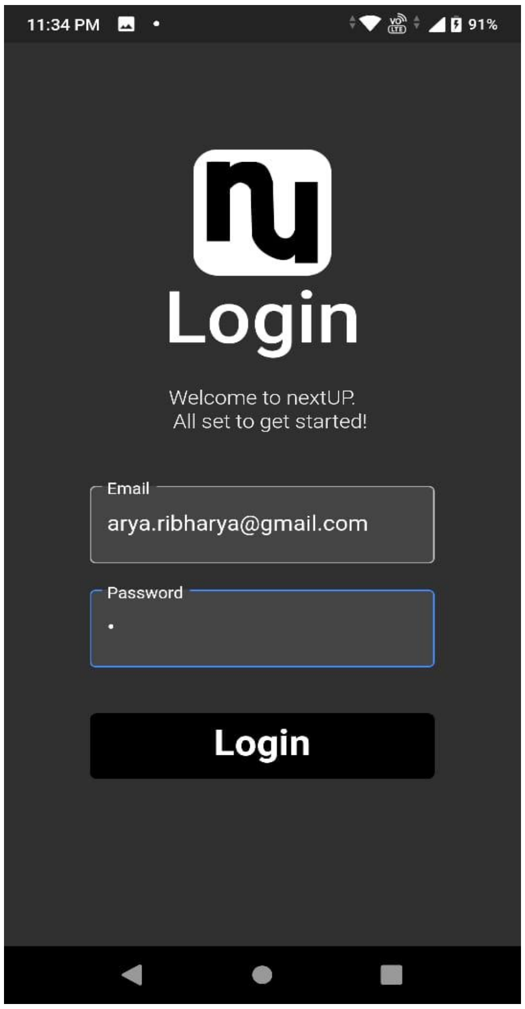


```
result = await project.aggregate([
    { $match: { projectID: projectID } },
    { $unwind: '$tasks' },
    { $match: { 'tasks.taskStage': stage } },
    {
        $group: {
            _id: '$tasks.taskTitle',
            tasks: {
                $push: {
                    taskTitle: '$tasks.taskTitle',
                    taskStage: '$tasks.taskStage',
                    description: '$tasks.description',
                    taskID: '$tasks.taskID',
                },
            },
        },
    },
]);
if (!result) {
    return response.sendError(res, 'Error occured while fetching task');
}
// console.log(result);
```

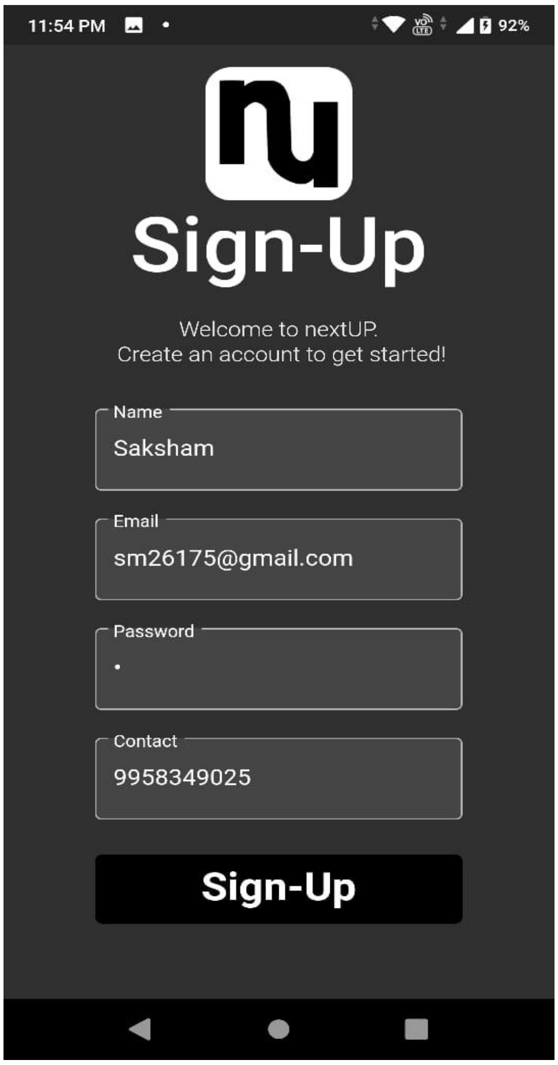
UI Screenshots

- **Main Screen**
- **Login Screen**





- **Sign-Up Screen**
- **Projects Screen**


11:54 PM 92%

Sign-Up

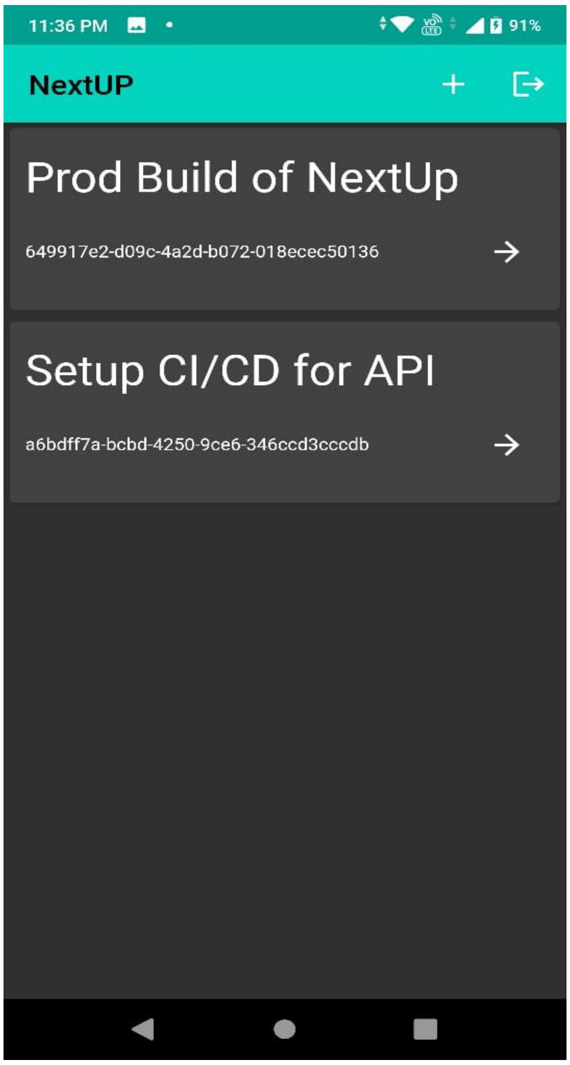
Welcome to nextUP.
Create an account to get started!

Name: Saksham

Email: sm26175@gmail.com

Password: •

Contact: 9958349025


11:36 PM 91%

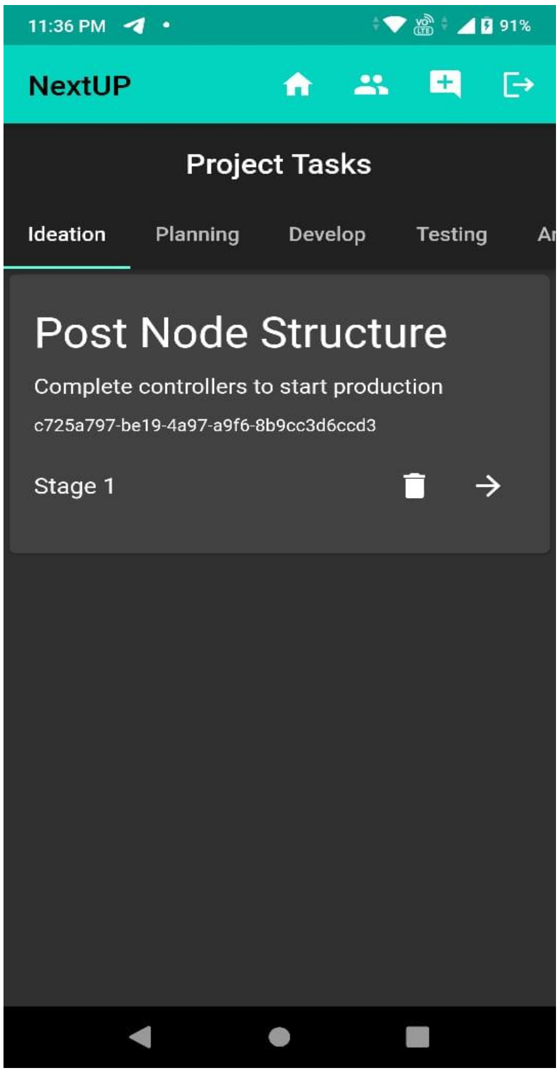
Prod Build of NextUp

649917e2-d09c-4a2d-b072-018ecec50136

Setup CI/CD for API

a6bdff7a-bcbd-4250-9ce6-346ccd3cccdb

- **Ideation Stage Tasks**
- **Planning Stage Tasks**

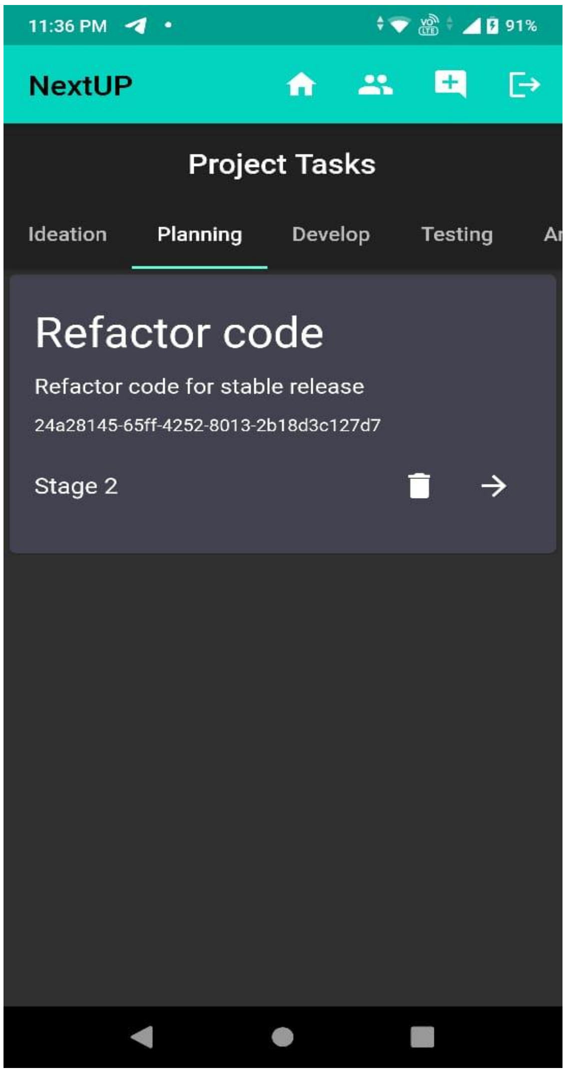


Post Node Structure

Complete controllers to start production

c725a797-be19-4a97-a9f6-8b9cc3d6ccd3

Stage 1



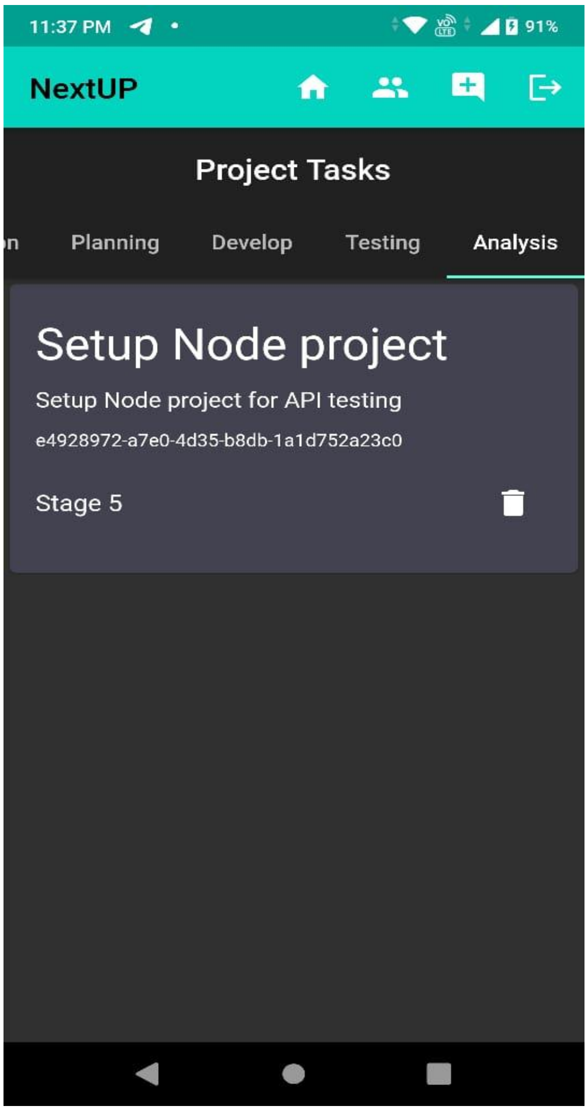
Refactor code

Refactor code for stable release

24a28145-65ff-4252-8013-2b18d3c127d7

Stage 2

- **Analysis Stage Tasks**
- **Add a Task**

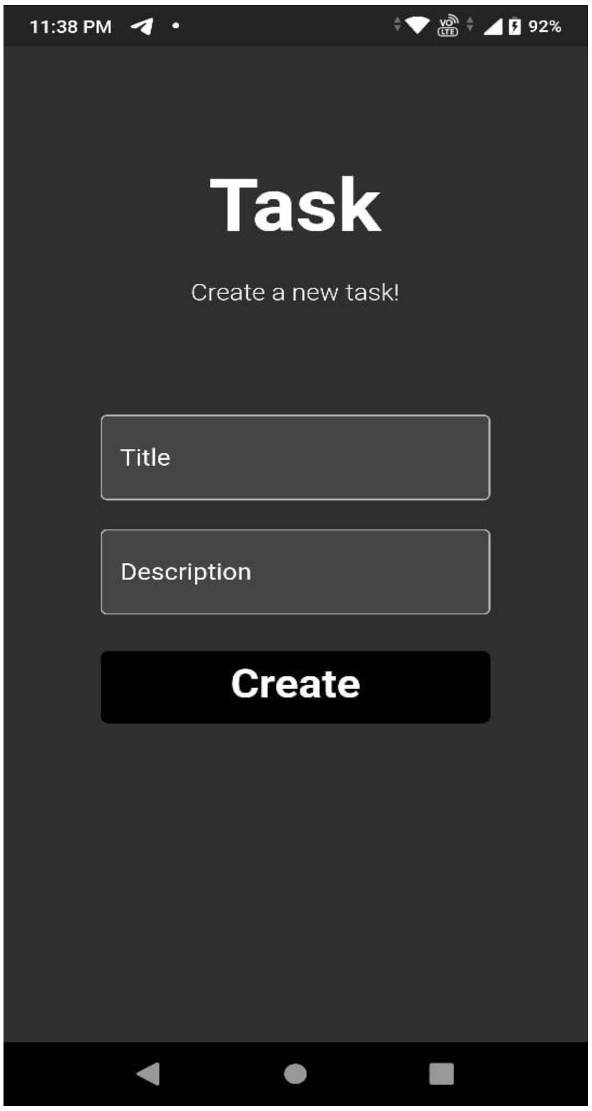


Setup Node project

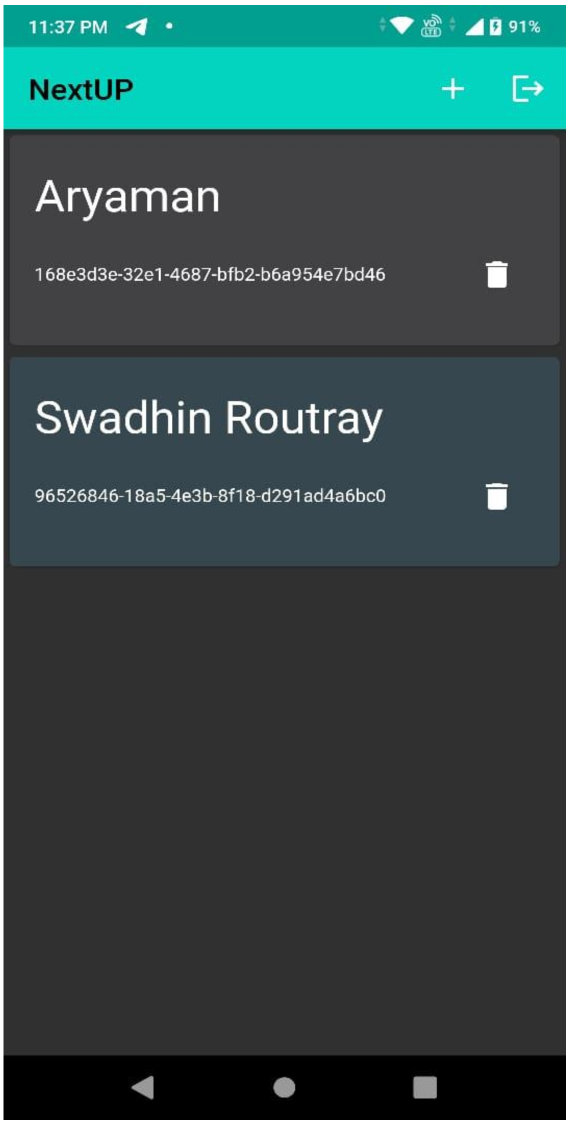
Setup Node project for API testing

e4928972-a7e0-4d35-b8db-1a1d752a23c0

Stage 5



- **Collaborators Screen**
- **Add a Collaborator**


11:37 PM 91%

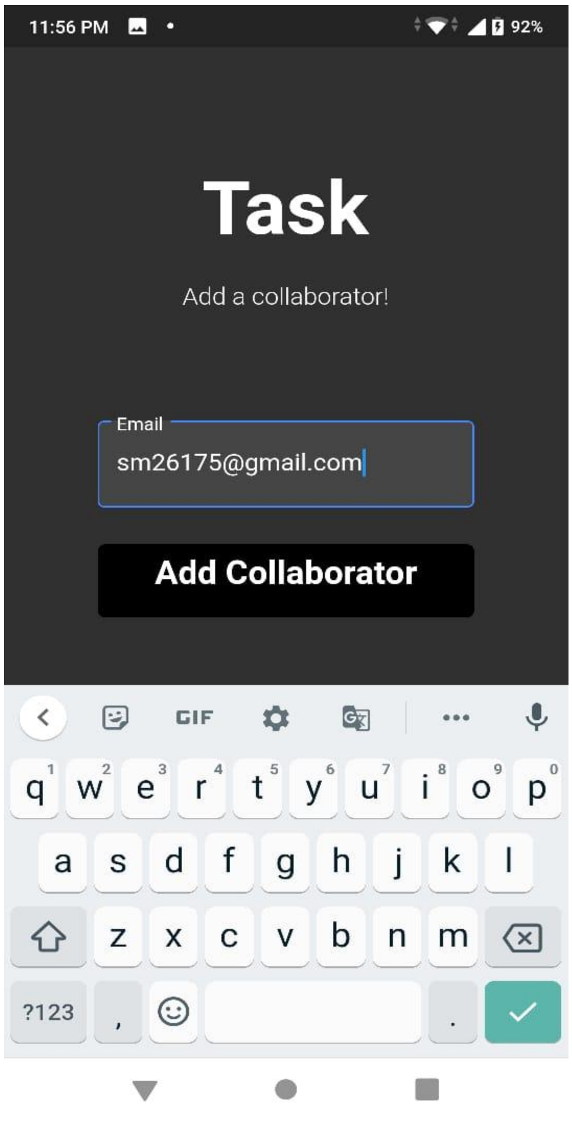
NextUP

Aryaman

168e3d3e-32e1-4687-bfb2-b6a954e7bd46

Swadhin Routray

96526846-18a5-4e3b-8f18-d291ad4a6bc0


11:56 PM 92%

Task

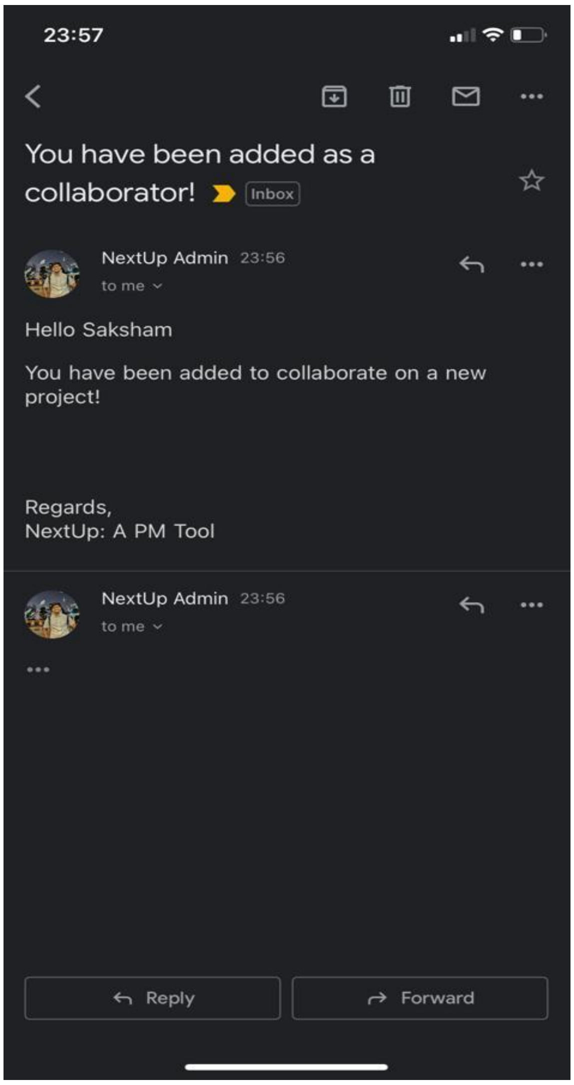
Add a collaborator!

Email

sm26175@gmail.com

Add Collaborator

- **Collaborator Added**
- **Updated Collaborator Screen**


You have been added as a collaborator!

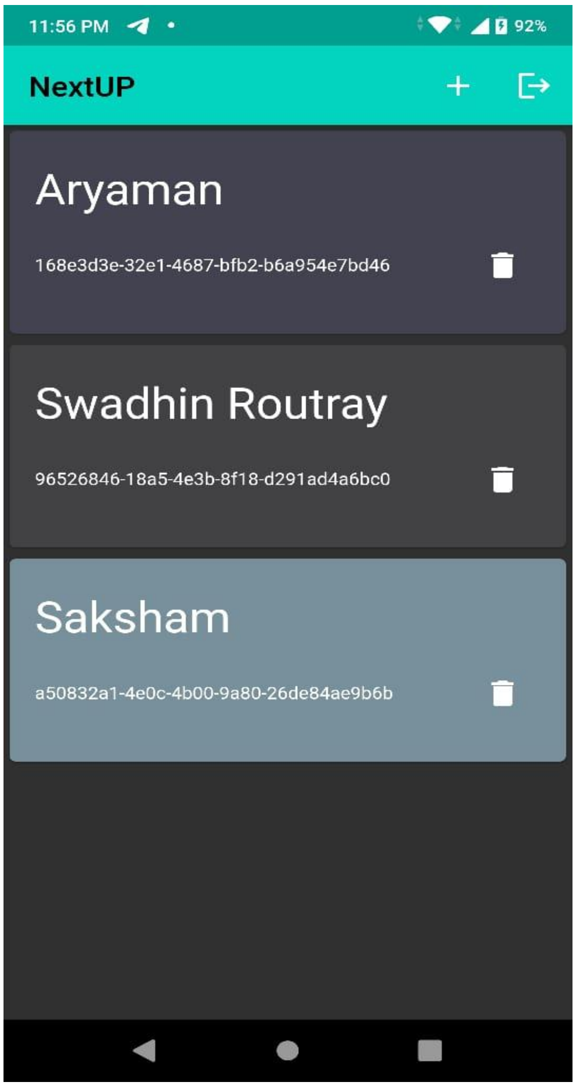
NextUp Admin 23:56
to me

Hello Saksham

You have been added to collaborate on a new project!

Regards,
NextUp: A PM Tool

NextUp Admin 23:56
to me

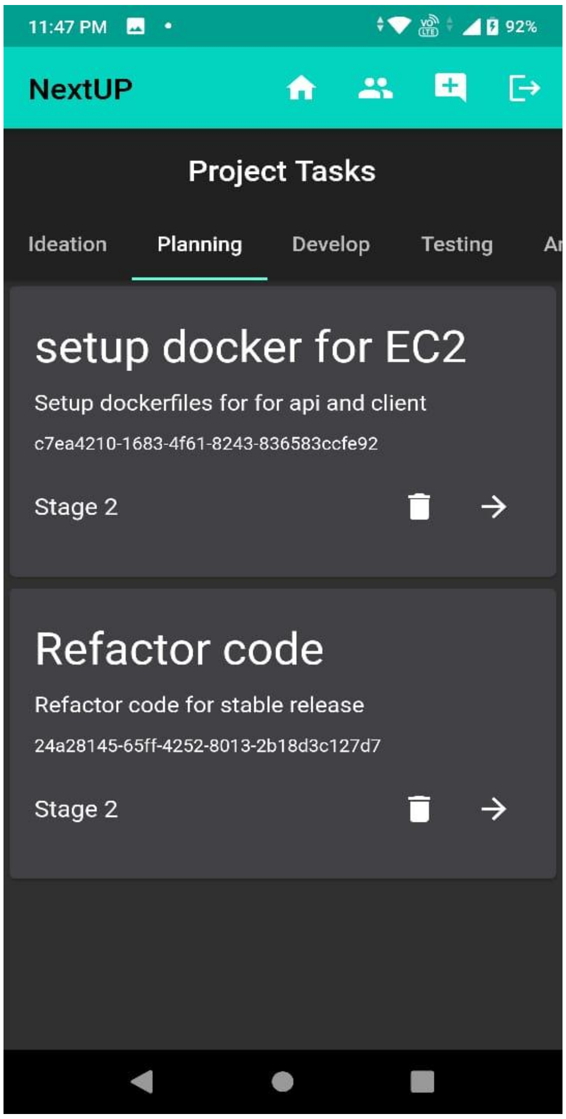

NextUP

Aryaman
168e3d3e-32e1-4687-bfb2-b6a954e7bd46

Swadhin Routray
96526846-18a5-4e3b-8f18-d291ad4a6bc0

Saksham
a50832a1-4e0c-4b00-9a80-26de84ae9b6b

- **Stage Transfer**
- **OTP for Stage Transfer**



11:47 PM 92%

NextUP

Project Tasks

Ideation Planning Develop Testing Ar

setup docker for EC2

Setup dockerfiles for for api and client

c7ea4210-1683-4f61-8243-836583ccfe92

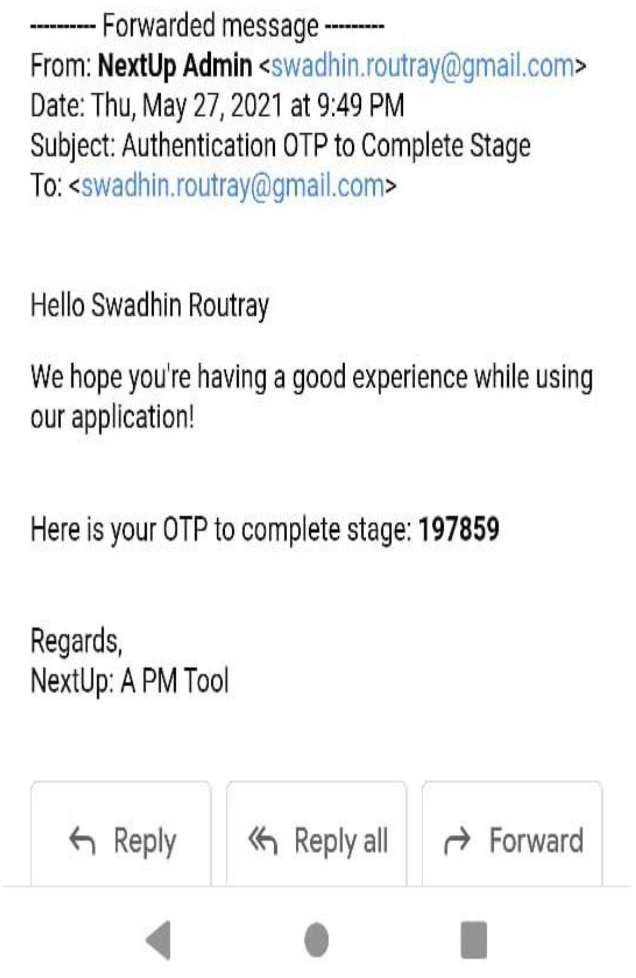
Stage 2

Refactor code

Refactor code for stable release

24a28145-65ff-4252-8013-2b18d3c127d7

Stage 2



---------- Forwarded message ---------
From: **NextUp Admin** <swadhin.routray@gmail.com>
Date: Thu, May 27, 2021 at 9:49 PM
Subject: Authentication OTP to Complete Stage
To: <swadhin.routray@gmail.com>

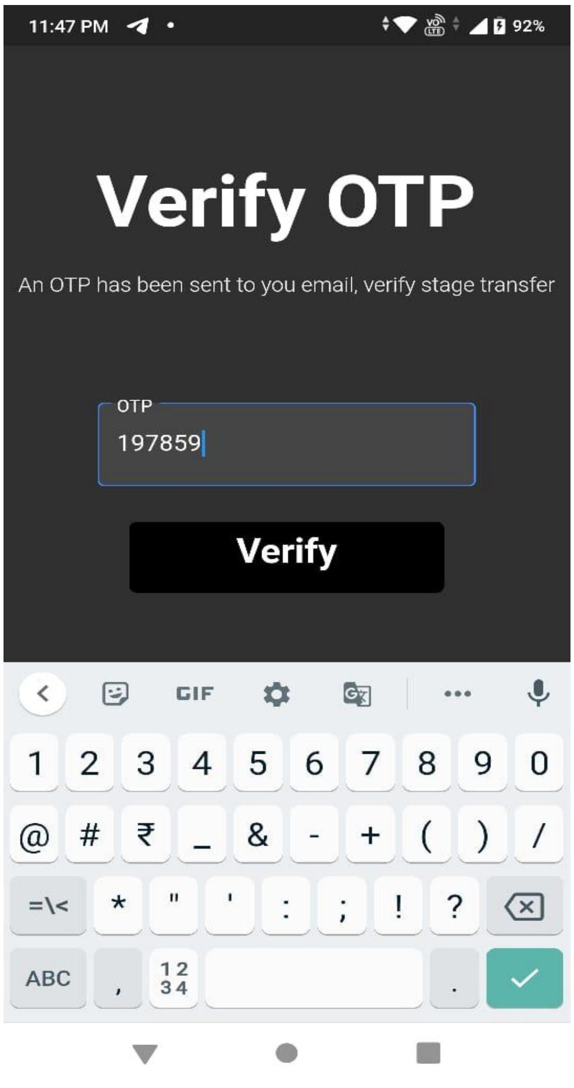
Hello Swadhin Routray

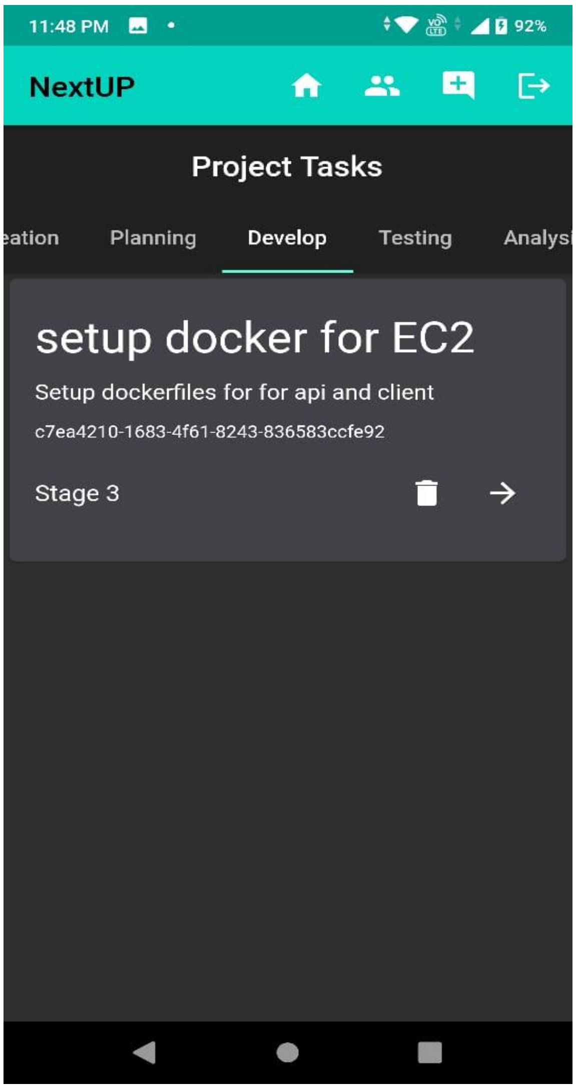
We hope you're having a good experience while using our application!

Here is your OTP to complete stage: **197859**

Regards,
NextUp: A PM Tool

- **OTP Verification**
- **Stage Transfer Successful**





RESULTS

The working prototype demo can be viewed on the link given below

Link to the Video

The results have been simulated using a Nexus 6 Emulator. The results have been published in the UI Screenshots section.

CONCLUSION

The NextUp application makes project management and task division extremely simple. It leads to easier collaboration over projects by multiple users. There are multiple applications like NextUp, however they can be further made more efficient. One of the major benefits of this application is task movement requires an acceptance in the form of an OTP and this makes it very important in two ways:

- Safeguards the task from force transfer into the next stage.
- Prevents premature task movement.

As a result it brings up efficiency in project management as well as helps keeping track of important tasks across all the stages.

FUTURE WORK

Application Based

- Attachment of Files related to the different tasks
- Budget Monitoring to prevent Project Overspend
- Publish APK for beta testing of the application
- Optimising Dart code for seamless screen transitions.

API Based

- Implementation of notifications using FCM(Firebase Cloud Messaging).
- Dockerizing API code for easier deployment.
- Setting up a pipeline for CI/CD for the API
- Implementation of controllers Budgeting for Project overspending.
- Implementation of CRON jobs for notifications on project deadlines passing.
- Deployment of the API to a cloud service provider to reduce load time on the Application.

Individual Tasks & Responsibilities

Swadhin Routray:

- Creation of a DB Schemas and integration with the API & custom API for implementation of create project, add collaborators, moving stages using OTP.
- Setting up a mailing server for sending OTP based emails using AWS SES.
- Sending notifications using Firebase Cloud messaging.
- Implementing 2FA functionality.
- Design and Implementation of adding a collaborator screen using.
- Developing the API Provider for backend access ease.

Aryaman Singh:

- Basic UI/UX design of the Application.
- Implementation of Individual Project's Screen.
- Implementation of individual Kanban Stage widgets for a particular project.
- Developing the tasks & progress screen.
- Design and Implementation of Login Screen.

Saksham Mehta:

- Implementation and Design of User's Projects Screen.
- Implementation of Project & Collaborator widgets card.
- Developing the OTP verification and Task addition screens.
- Design and Implementation of Login & Signup Screen.

REFERENCES

1. Dart Language Documentation: https://dart.dev/
2. Flutter Documentation: https://flutter.dev/docs/get-started/codelab-web
3. Flutter Widgets: https://flutter.dev/docs/development/ui/widgets
4. Node.js Documentation: https://nodejs.org/
5. Dio Documentation: https://pub.dev/packages/dio
6. MongoDB Documentation: https://www.mongodb.com/