MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

Computer Engineering Program

CNG 495

CLOUD COMPUTING

FALL 2025- TERM PROJECT PROPOSAL

Team Members:

Nazlıcan Taviş- 2751741

Nisa Sağdıç- 2751691

Fatih Demirbilek – 2526234

WashMate

The aim of this project is to develop a cloud-based Laundry Management System that enables students to efficiently manage their laundry schedules through a user-friendly web site. The students will be able to register in our system with their student number, email, name and surname. Their phone number will be used to send notifications. Then they will be able to log in to view the dashboard of users. In the user dashboard, they can view the current status of all available washing and drying machines, whether they are available, in use, or disabled in a table format. Users can reserve available time slots so they will not waste their time waiting for a machine or creating conflicts to reserve. After they went to wash their clothes, they will state that in the interface. If a user books a time slot but fails to attend three times, the system will issue a warning. Additionally, if a user books a slot but does not show up, the slot will be reopened after five minutes.Every student will be able to see booked time slots. When they booked available time, when the machine finishes it washing they will receive a notification via their email.

The application is designed to make the laundry process more organized and accessible within students and dry cleaner service providers by offering real-time machine availability updates and online booking features, it significantly improves user convenience and resource utilization. In addition**,** users can change the machine status as disabled directly through the system, so admin can see those problems and solve them immediately.

Administrators will have access to an admin dashboard where they can add or remove machines. Also, the admin will have right to reject or approve the student registrations. They will also be able to see machine statistics, such as which machine has how many drying/washing cycles, etc.

The Smart Laundry System integrates cloud technologies to deliver a reliable, and efficient solution for students who use common machines. By hosting the data and services in the cloud, large amounts of user and machine data can be managed easily and the system ensures high availability.

Implementation

Frontend - Client-Side Web Application (HTML, CSS, JavaScript)

The WashMate system is a client-side web application that uses HTML, CSS, and JavaScript. First, we planned to use AWS, but since Firebase provides integrated backend services, authentication, database access, and cloud functions that can be triggered directly from the frontend, we switched to Firebase.

The frontend interface will manage:

• User and admin login and registration

• User operations such as booking laundry time slots, viewing the information about dry clean service, and changing machine status

• Admin operations such as approving users and managing machines

Firebase's JavaScript SDK will be used to communicate securely with the backend cloud services.

Cloud Authentication Service – Firebase Authentication

User authentication will be provided by Firebase Authentication. The service supports secure login and registration using email and password. Verification emails, password reset operations, and token-based login are all handled automatically by Firebase; thanks to this, there is no need for custom, specific security. User roles such as admin or student will be stored using Firebase Authentication custom claims. This allows the system to distinguish between administrators and regular users when accessing protected pages.

Cloud Database Service – Firebase Firestore (NoSQL)

Google Firebase Firestore will be used as the primary cloud-hosted NoSQL database. The database will store the necessary WashMate App's data, and Firestore provides real-time data synchronisation, such as allowing machine availability changes to instantly appear on all connected clients. Thanks to this, there is no need for manual page refreshes, and it is a fully dynamic dashboard. The database operates under the Database as a Service (DBaaS) model, providing scalability and high availability without requiring manual server management.

Cloud Notification Service – Firebase Cloud Functions + Email Service

At first, we planned to use phone numbers and SMS, but it is easier in Firebase to use the email service. The system will use email notifications triggered via Firebase Cloud Functions. The system sends an email in 2 main cases: when a washing or drying cycle is completed, Cloud Functions automatically detects the status change in Firestore and sends an email to the user. The second case is when the student didn't attend the booked time slot 3 times.

To make the email service work, these functionalities will be used:

• Cloud Functions (serverless backend for event triggers)

• Nodemailer or an external email provider (e.g., Gmail API, SendGrid)

Cloud Hosting Service – Firebase Hosting

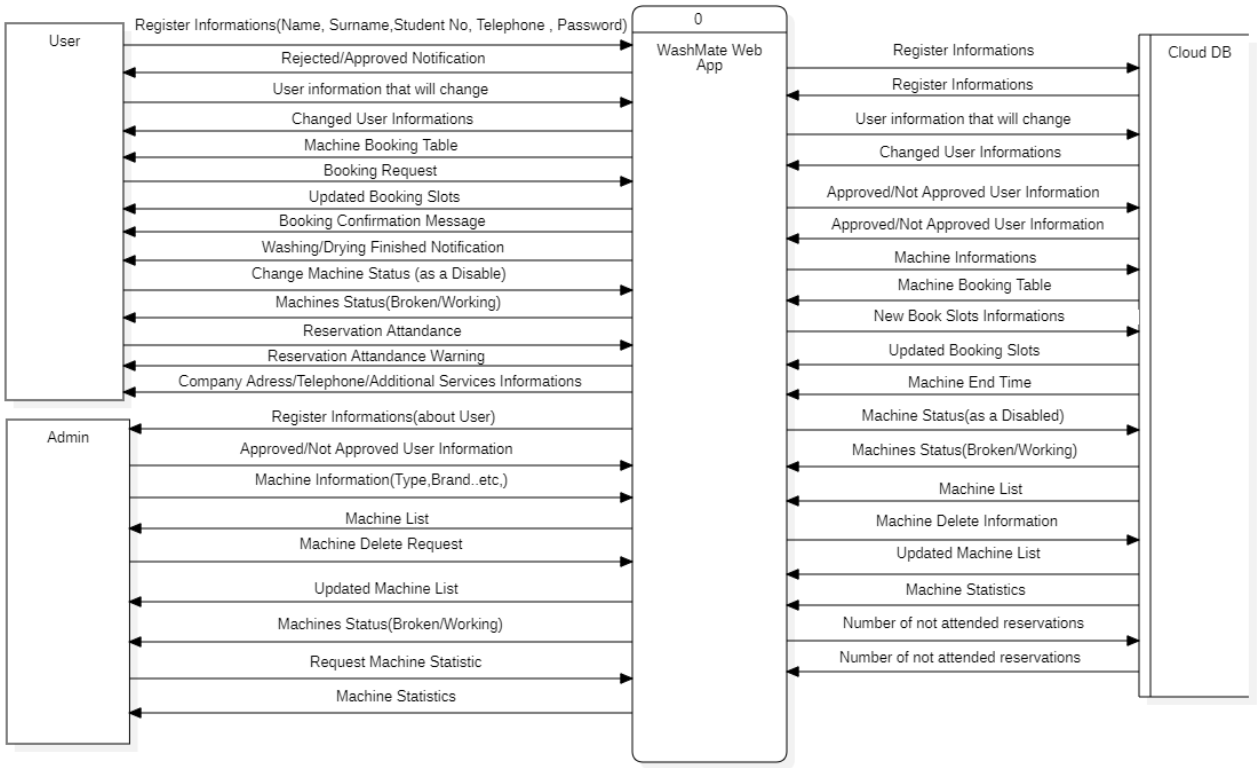
The web application will be deployed using Firebase Hosting. This service delivers the frontend files (HTML, CSS, JS) through a global Content Delivery Network (CDN), ensuring fast load times and high availability. Firebase Hosting supports HTTPS by default and integrates directly with Authentication and Firestore, making deployment straightforward.

The system will be accessible through a publicly available Firebase Hosting URL. This is the URL:**https://washmate-227cf.web.app/**

Diagrams

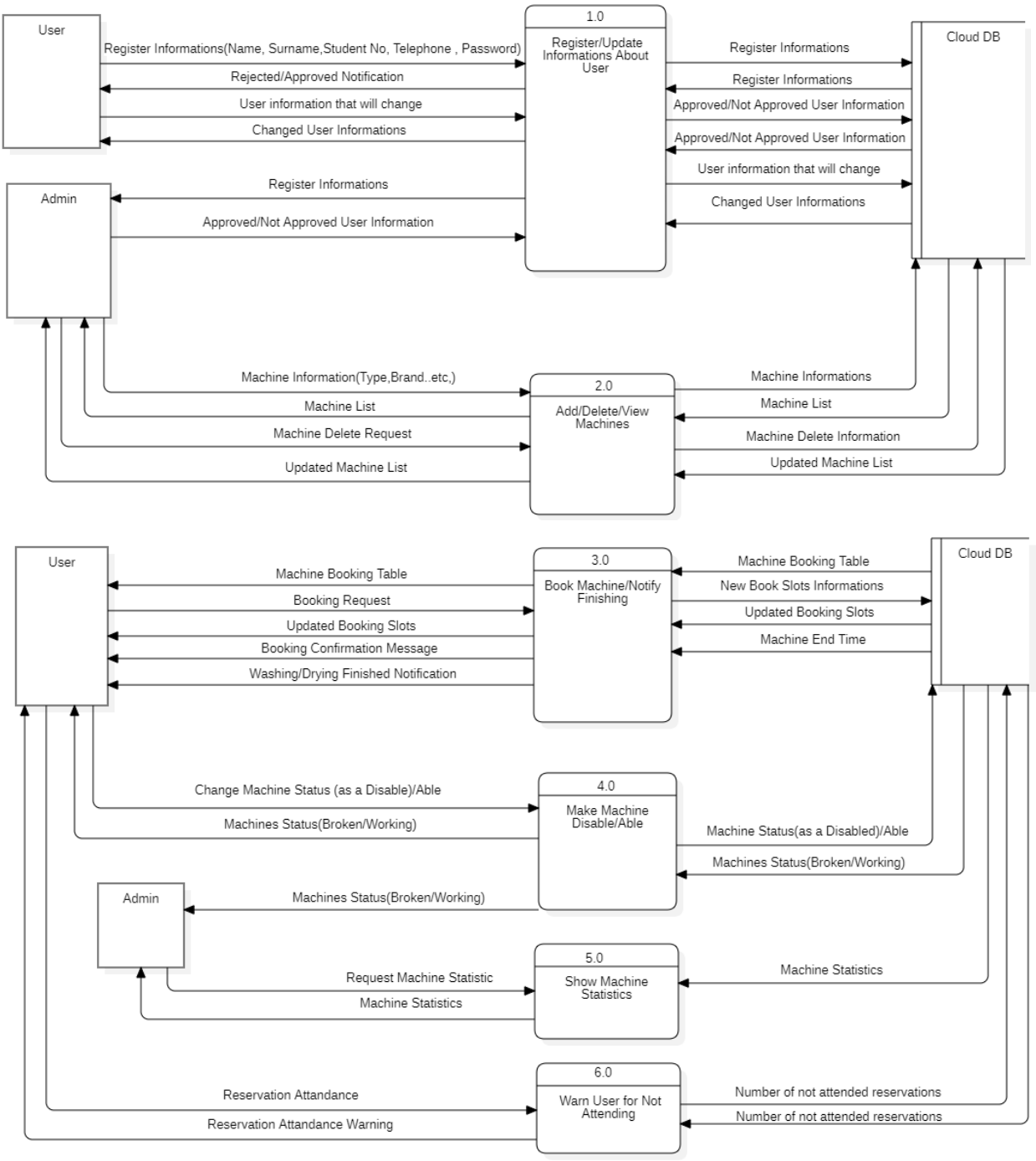
Data-Flow Diagram:

0-Level:



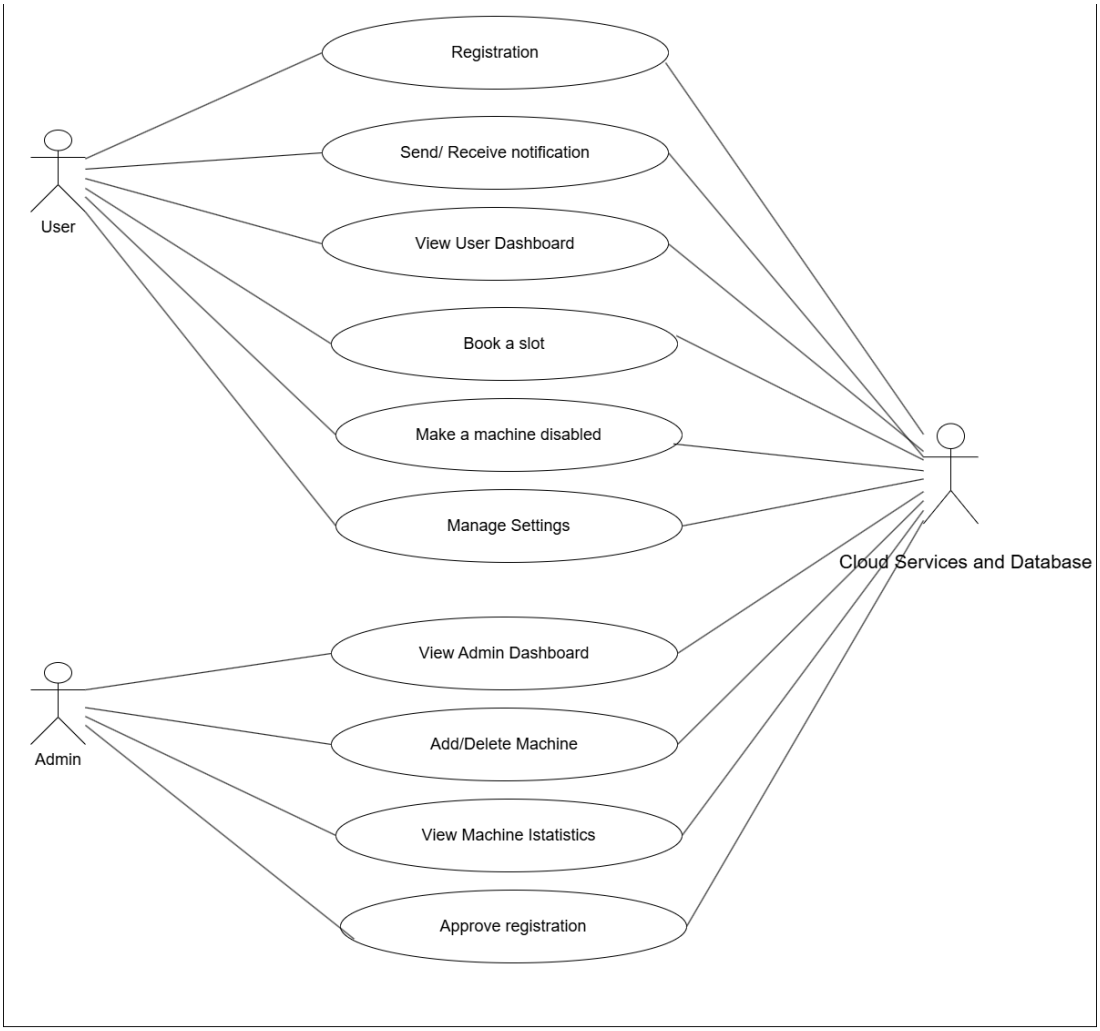
In the 0-Level data flow diagram, we illustrated the total data flow between the application, user, administrator, and cloud database. These data flows also explain the app's functions. We haven't shown cloud services separately here; we've assumed they're part of the app. We anticipate more detailed and extensive data flows during the implementation of our project, but we've illustrated the major functions in this diagram.

1-Level:



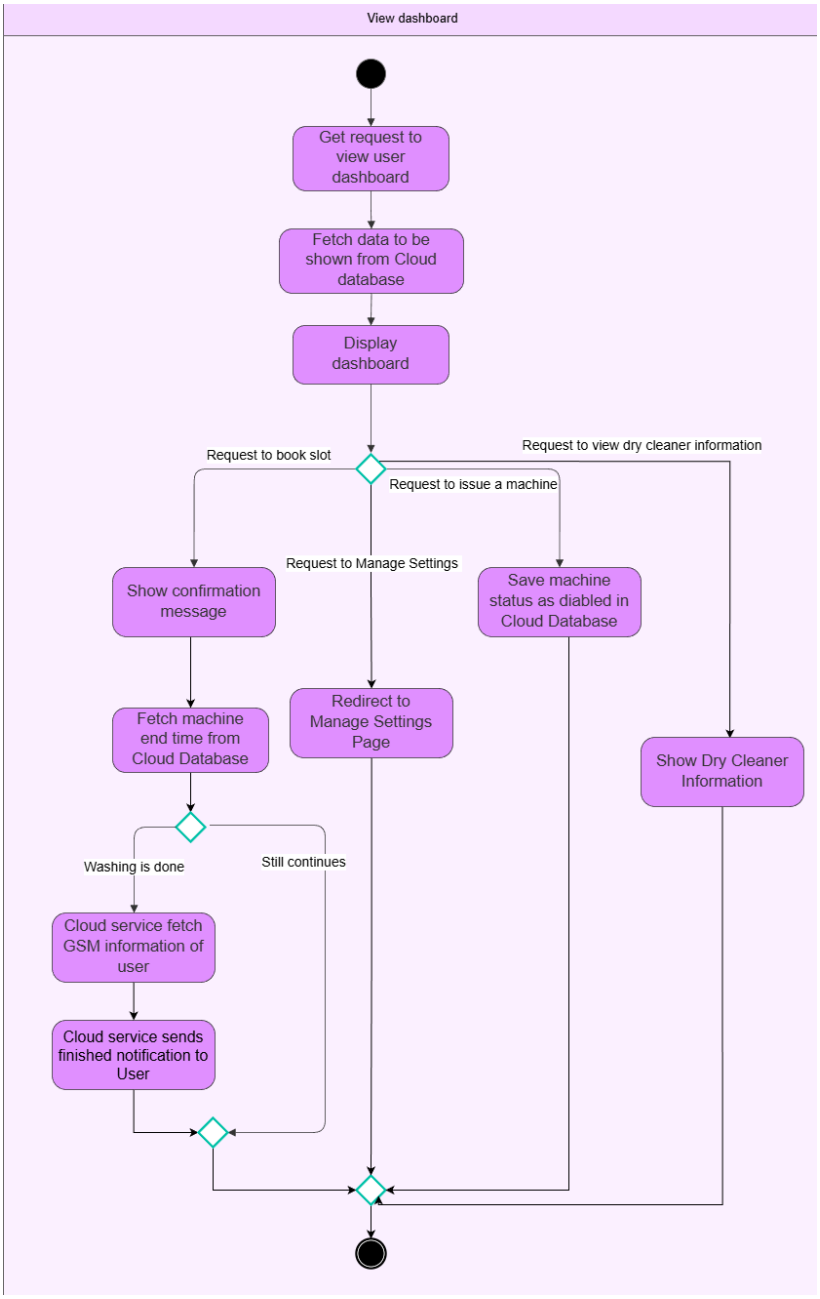
In 1-Level data flow diagrams, we displayed the data flow of the app's main functions separately. We combined functions that serve the same purpose into the diagram. This allowed us to display the data required for each function in greater detail.

Use Case Diagram:

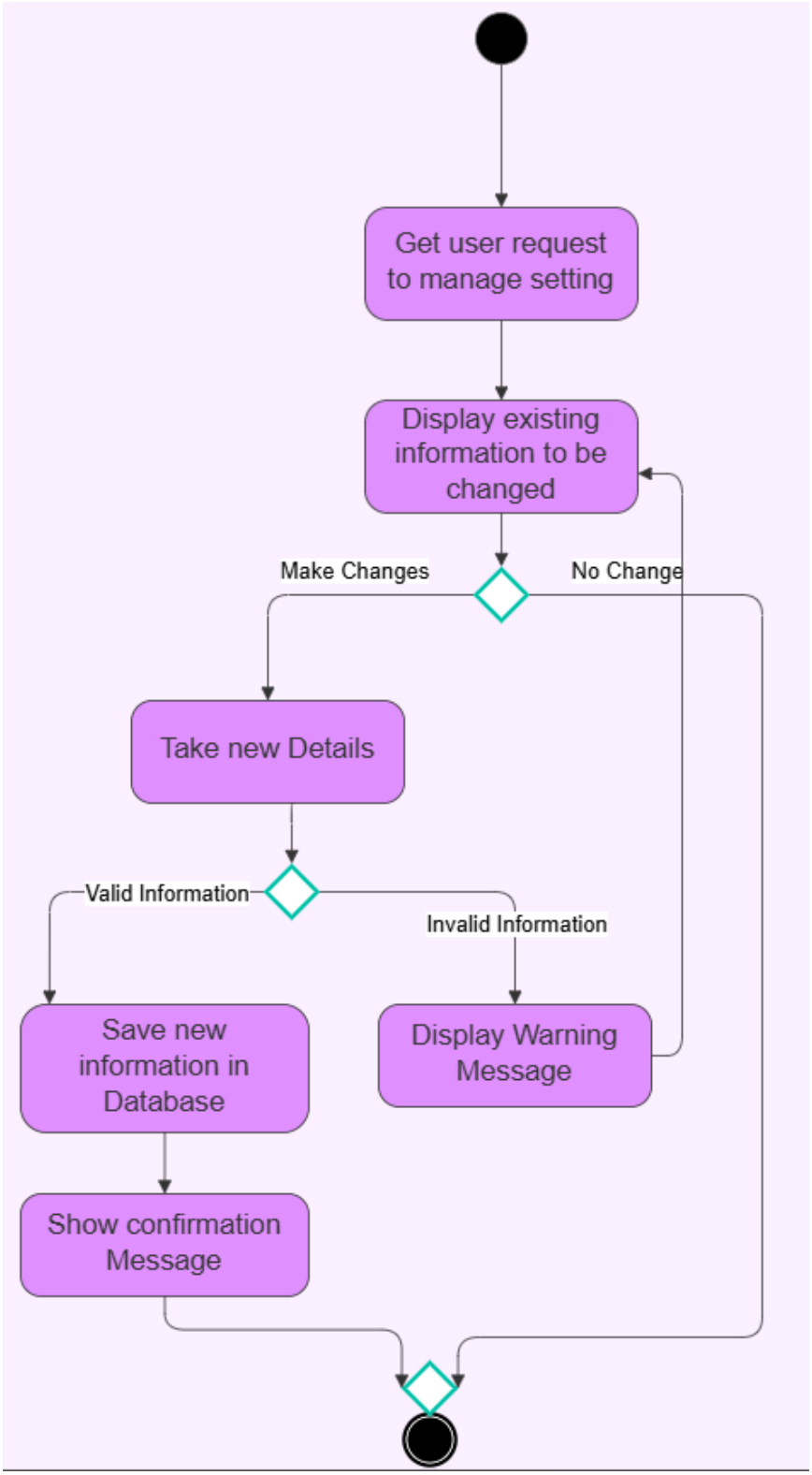


We explained the admin and user functions in the WashMate app using a use case diagram. We also represented the cloud database and services together.

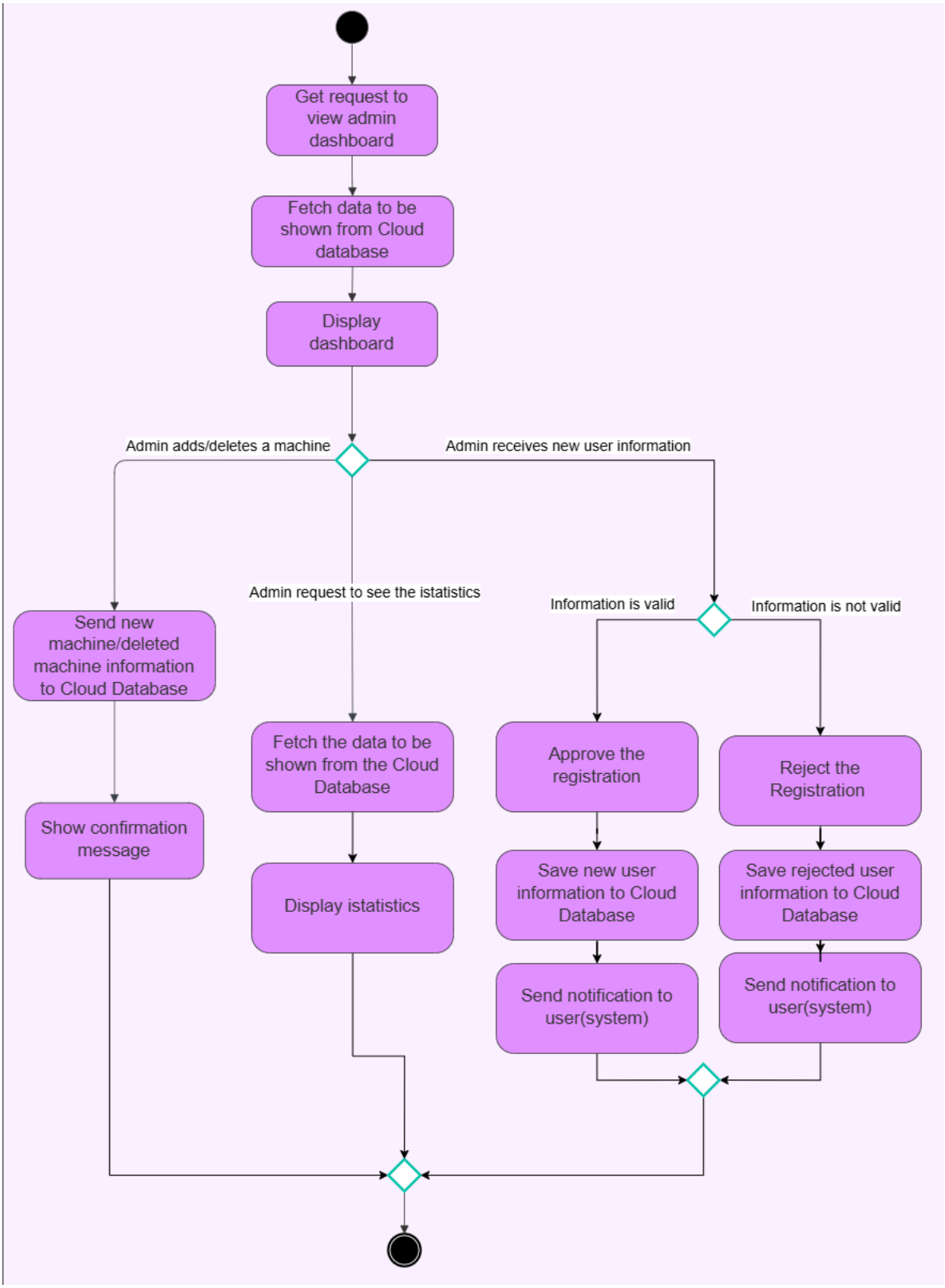
Activity Diagrams:



This activity diagram clearly shows the view user dashboard. In user dashboard user will see the booking table and have option to reserve a time. Also, they will be able to see dry cleaner information when they click on the info icon. After that user will get notification at the end of the washing cycle. Other operation is user will have option to manage their settings, it is another page which will be explained in next activity diagram. And in the dashboard when they have issue with the machine they will be able to mark that machine as disabled.



This diagram illustrates manage settings page which user will have option to change their login information. They can change their information or not so we have two options for that when they want to change we will save new info into database and show confirmation message to user.



This activity diagram is for admin viewing dashboard. All operations tried to be explained in the same activity diagram. Admin have option to add or delete a machine. They will also have option to view the statistics fort he machines. Also, the admin can approve user registration.

Milestones Achieved

Weeks	Milestones
Week 1–Week 2-Week 3 (29 SEP-19OCT)	Introduction to the cloud, formation of the team, determination of the project idea, creation of project details, writing of the proposal
Week 4 (20 OCT-26OCT)	Researching AWS technologies and learning cloud technologies.
Week 5 (27 OCT-2NOV)	Modifying the proposal report based on feedback, researching Firestore cloud services and comparing them with AWS.
Week 6 (3 NOV-9 NOV)	Deciding to use Firebase, learning the basics of the Firebase platform, and having team members practice using a template project.
Week 7 (10 NOV-16 NOV)	Creating the Washmate project, testing student login and machine list processes with simple UI, learning Firestore NoSQL database principles.
Week 8 (17 NOV-23 NOV)	Implementation of student register/admin register and login processes, further development of the UI.
Week 9 (24 NOV-30 NOV)	Implementing the operations of adding and deleting machines (admin), viewing the machine list and status (user), finalizing the UI, writing a progress report

Work Completed

All Member:

- Researching the Firebase platform and determining its suitability for the project.
- Learning and practicing the features of the Firebase platform was done by all members through a shared template project.
- Writing progress report.

Nazlıcan:

- Correction of the proposal report.
- Sample implementation of adding and deleting machines for the admin (Firestore database and UI).
- Streamlining and finalizing the UI.
- Opening Github for the project

Nisa:

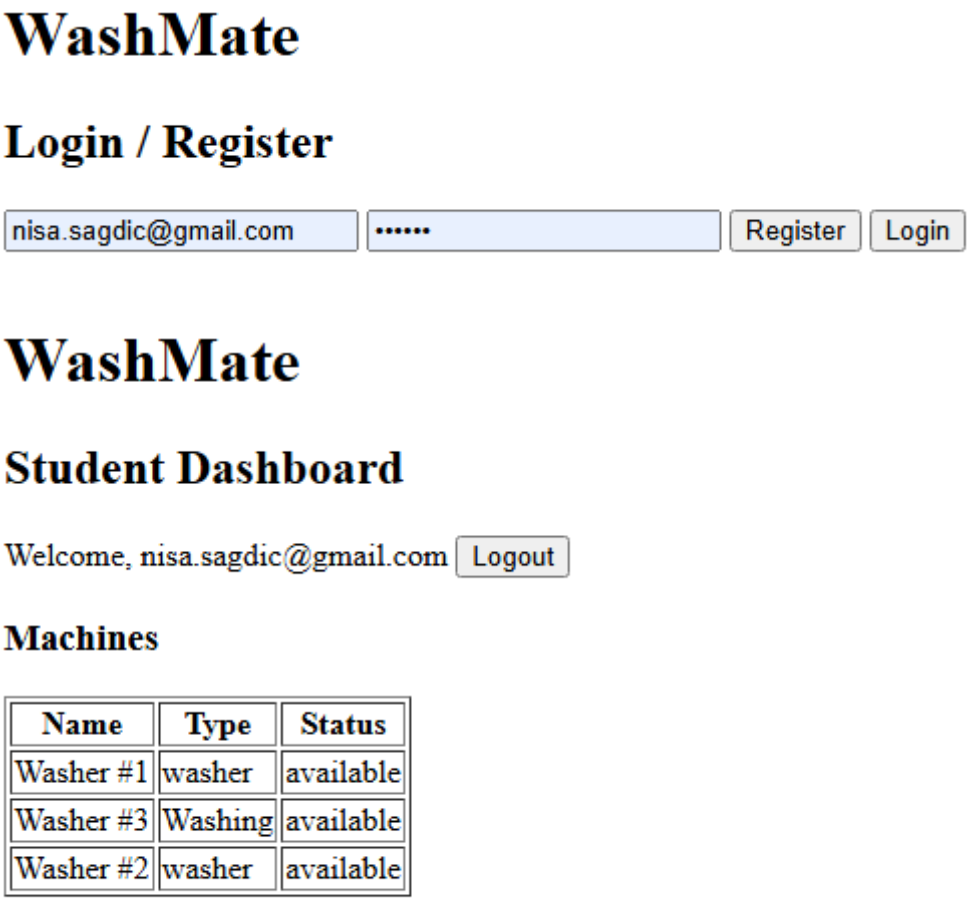
- Correction of the proposal report.
- Creating student and admin registration forms (UI and FireStore database).
- Improvement of the UI of the first version.

Fatih:

- Initial creation of the WashMate project.
- Initial creation of student register/login operations with basic UI and firebase database (first version).
- Creating an example of a machine list display process.

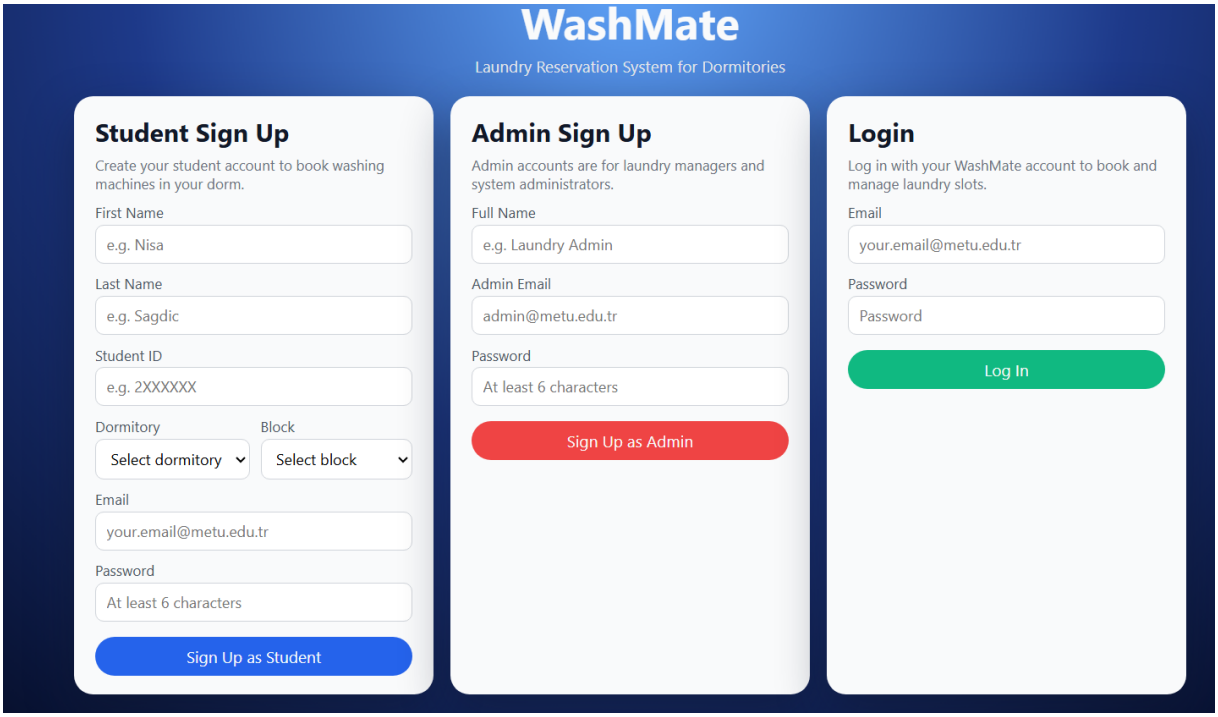
WashMate App Inıtıal Version (FATİH)

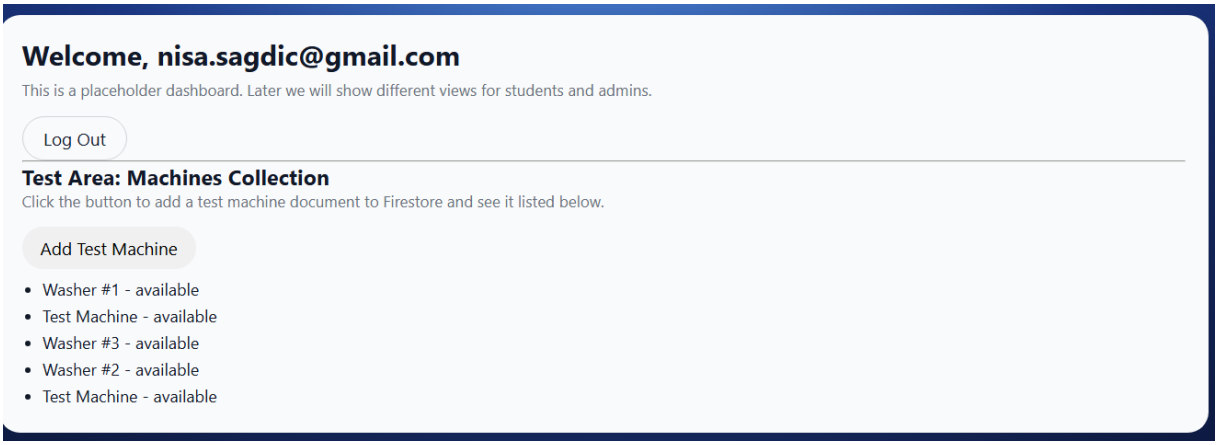
In this version, the WashMate App Project was initialized. A simple student registration/login and sample dashboard were demonstrated.



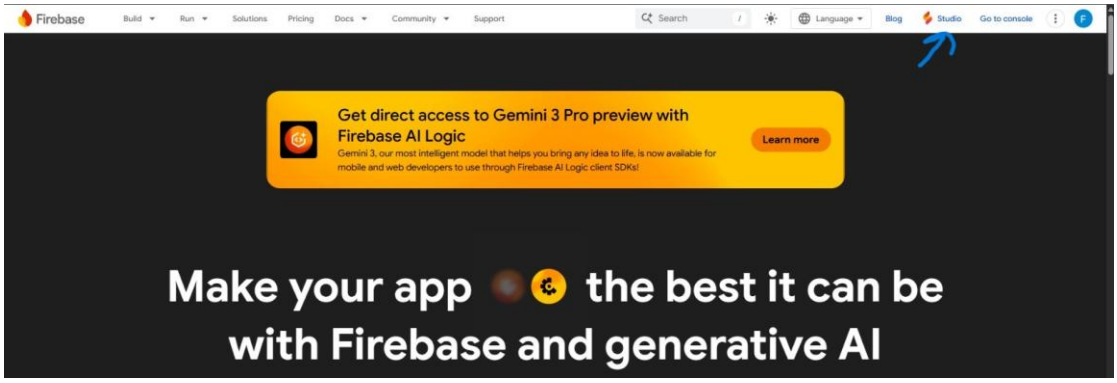
WashMate Improved Version (NİSA)

In this version, student/admin registration forms have been created and enhanced to be compatible with the data in our project. The appropriate data has been added as a FireStore database field. The machine addition process for the admin was initialized and tested.





Tutorial for Cloud Technologies:

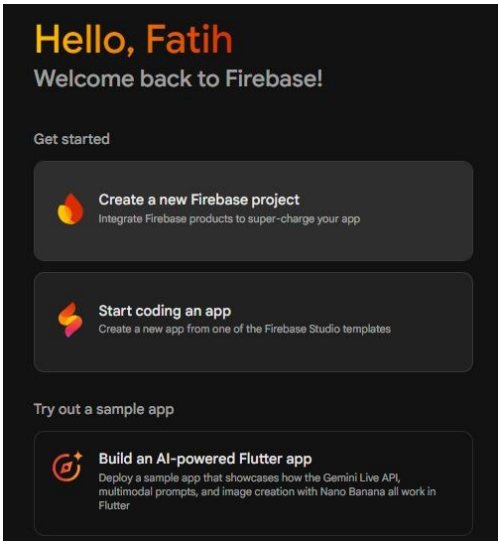


1. Go to the Firebase Homepage

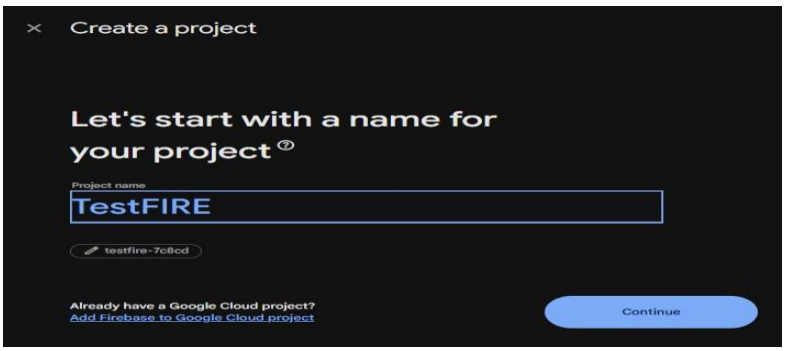
- This is the main landing page where you start using Firebase services.

-Open Firebase Console

2. Creating a New Firebase Project



-To create a new project, click **“**Create a new Firebase project.”

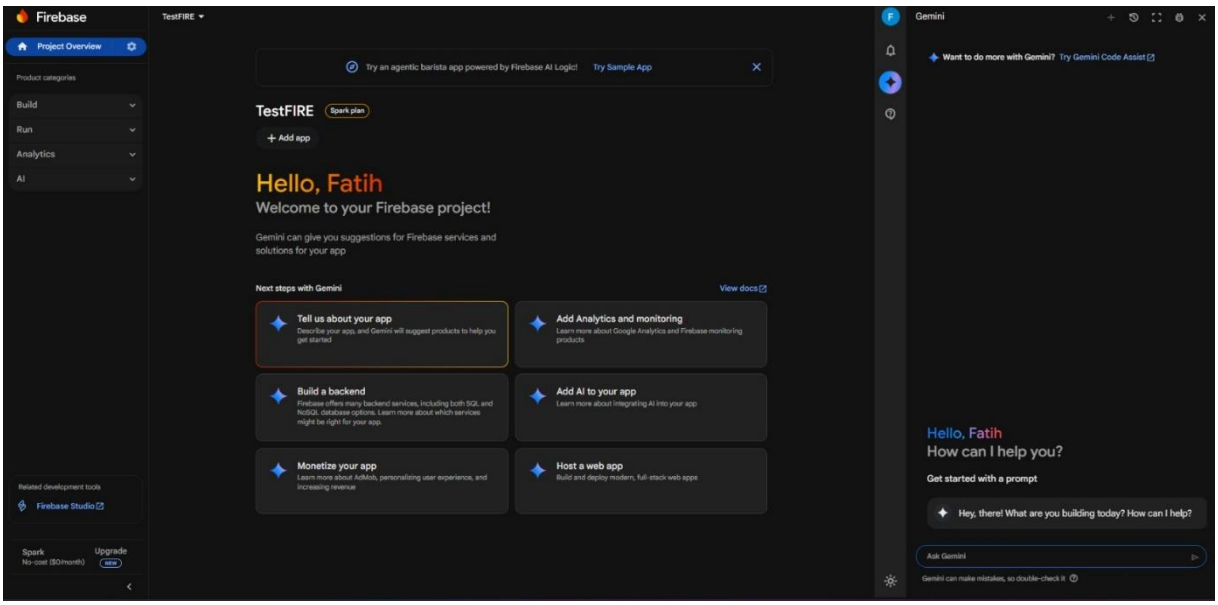


-After entering the name, click Continue.

-This will take you to the project setup wizard, where you can configure and initialize your Firebase project.

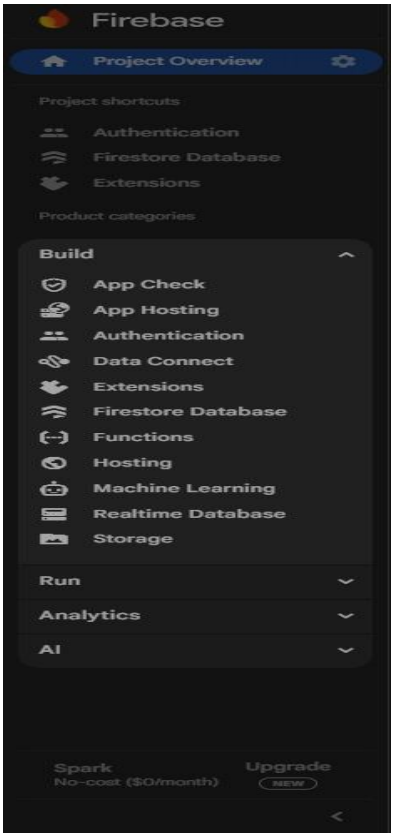
-After completing all required configuration steps, Firebase finishes setting up your environment.

3. Firebase Project Console Overview

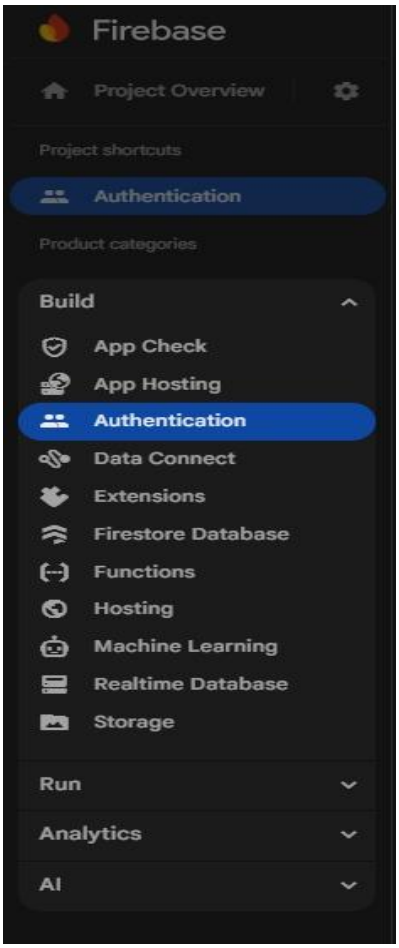


-This is the main Firebase project console, where you manage everything related to your project.

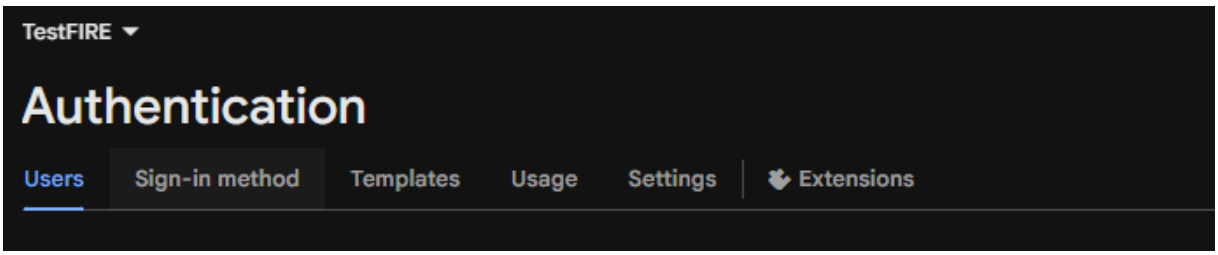
-The center section provides shortcuts for actions like building a backend, adding analytics, integrating AI, or hosting a web app.



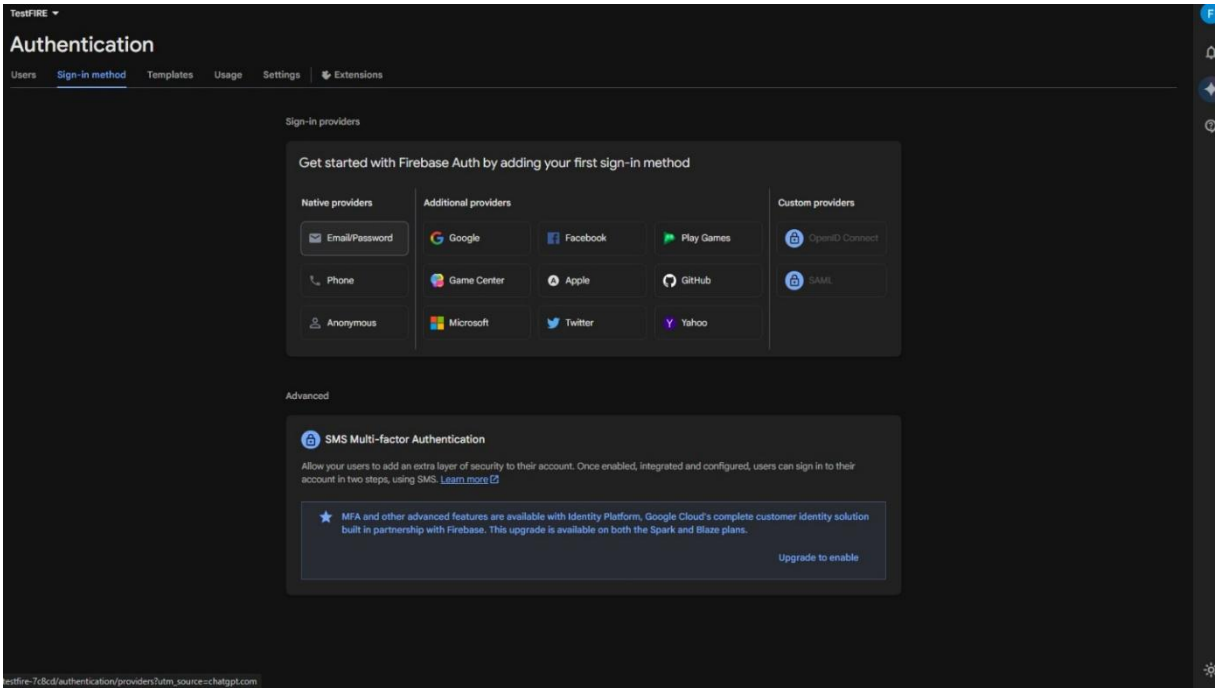
-From the left sidebar, you can access core features such as Authentication, Firestore Database, Storage, Hosting, and more.



-To enable user login and account management, go to the Authentication section from the left sidebar.



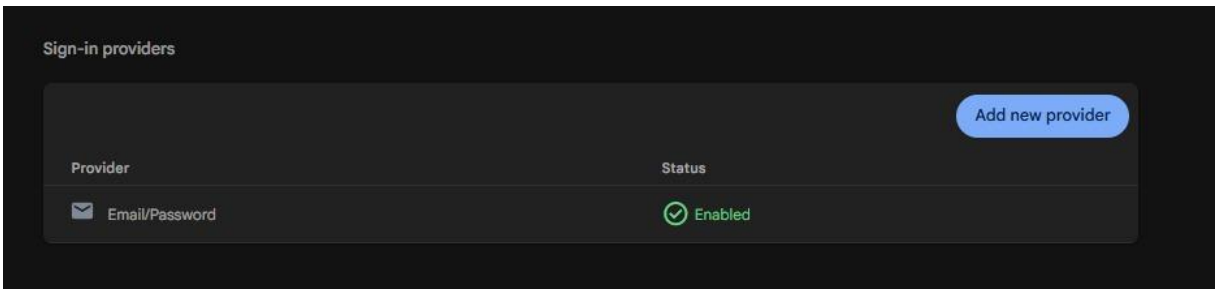
-Firebase Authentication allows you to add secure sign-in methods such as Email/Password, Google, Phone, and more



-This page allows you to configure how users will sign in to your application.

-Firebase Authentication supports multiple options such as:

- Email/Password
- Phone
- Google, Facebook, GitHub, Apple, and more
- Anonymous sign-in

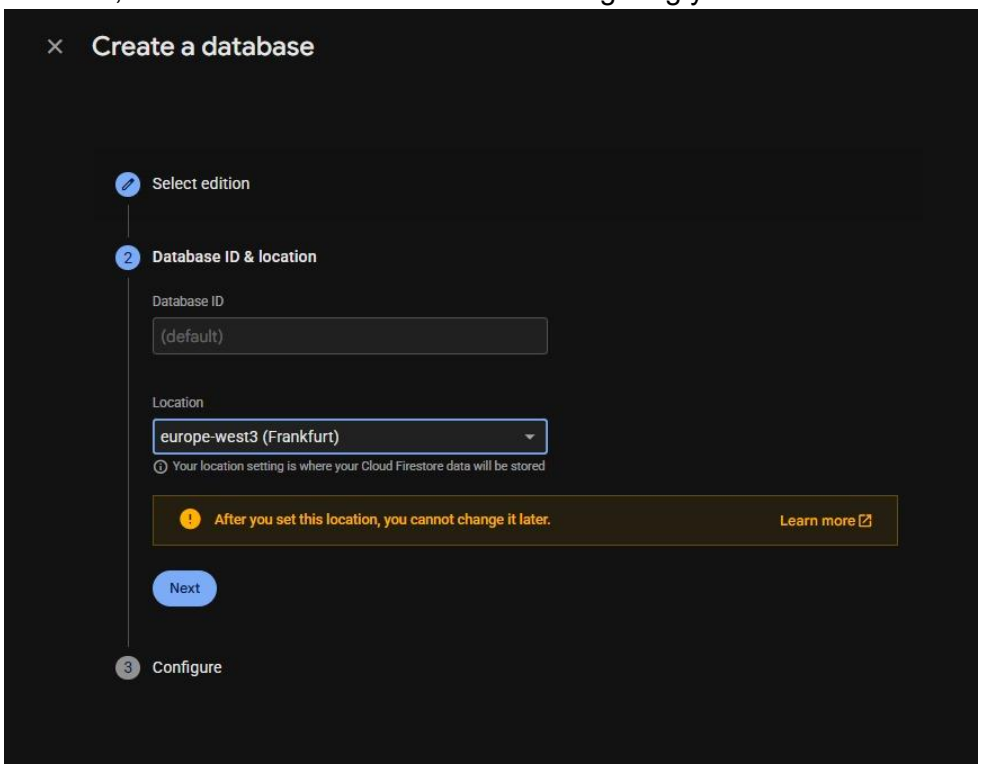


-Just click the provider you want to use and toggle it **ON** to activate it for your project.

4.SettingUp Cloud Firestore

-Cloud Firestore is Firebase's NoSQL database used for storing and syncing app data in real time.

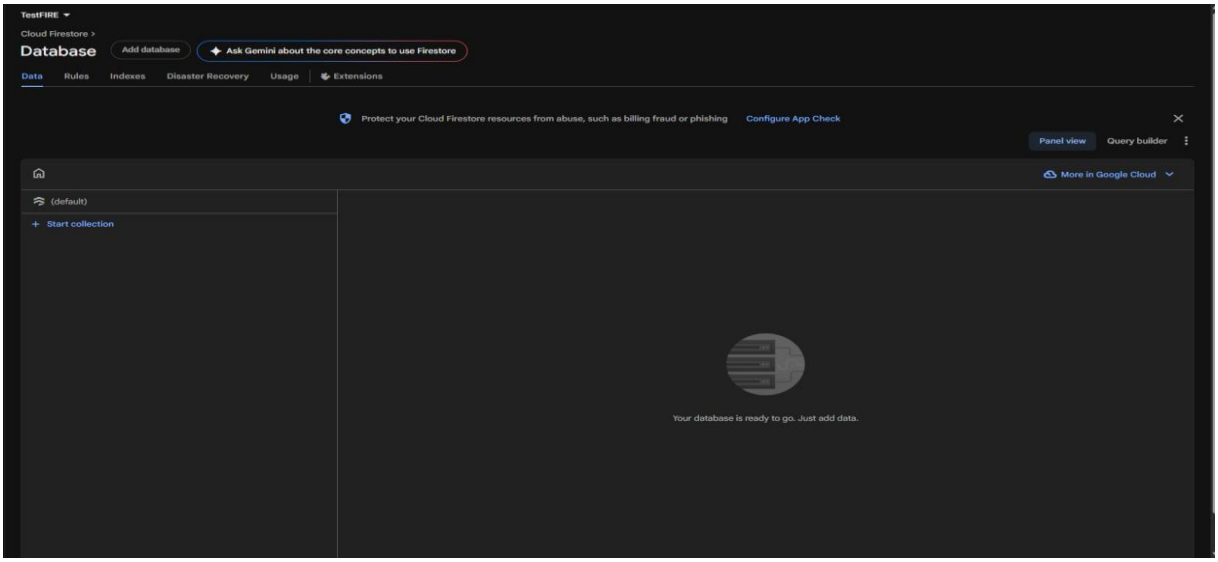
-From this screen, click "Create database" to start configuring your Firestore database.



On this screen, you select where your Cloud Firestore data will be stored. It's recommended to choose a region **closest to you:**

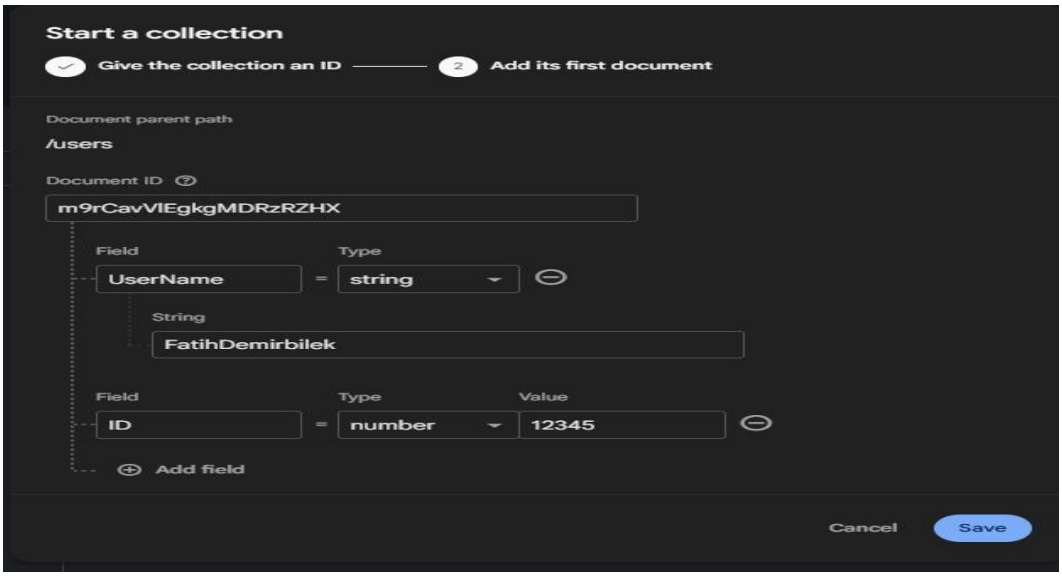
- Faster read/write performance
- Lower latency
- Better overall app responsiveness

-Firestore offers features like real-time updates, scalable storage, and powerful queries—perfect for apps that need constantly updated data.



-After creating the database, you are taken to the Firestore Data Dashboard.

-This is where you manage all collections and documents in your Firestore database.

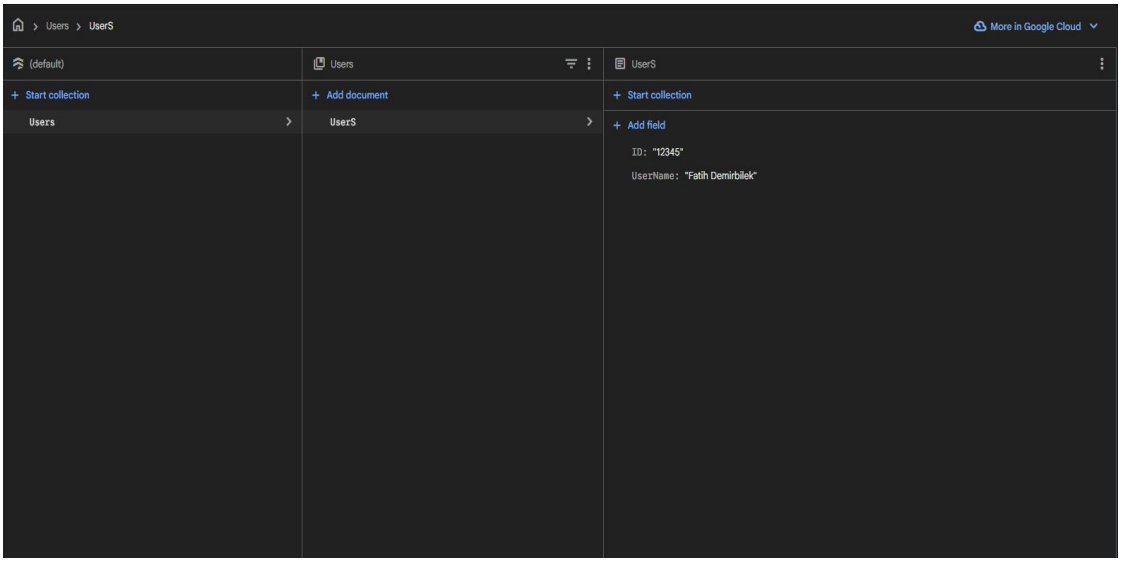


-Here we create a new Firestore collection—for example, “users”

-Each collection contains documents, and every document holds fields (key–value pairs).

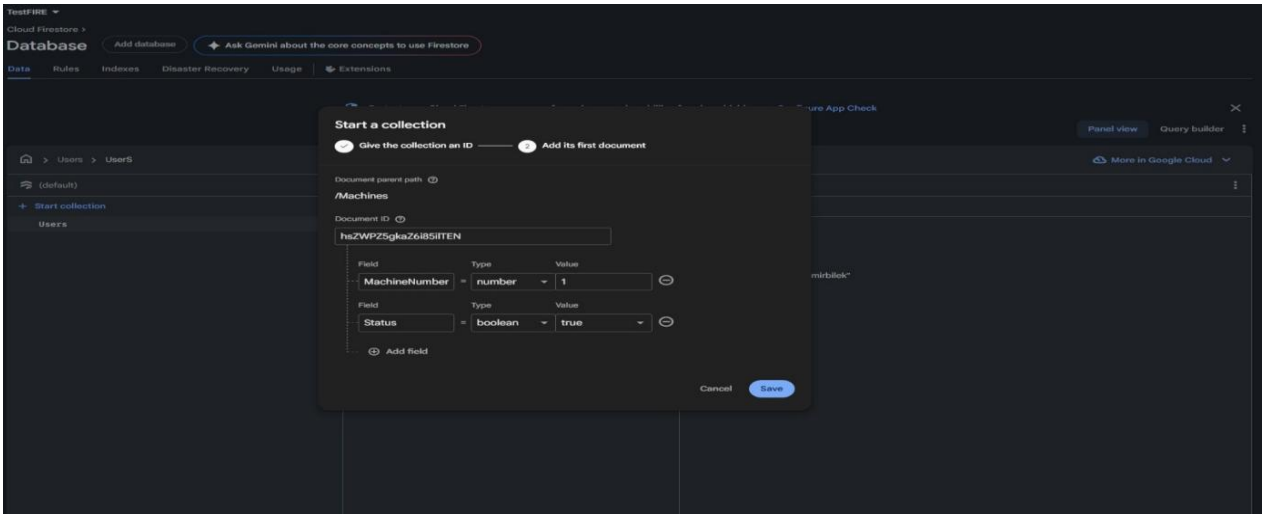
-Choose the field type (string, number, boolean, etc.) and enter its value.

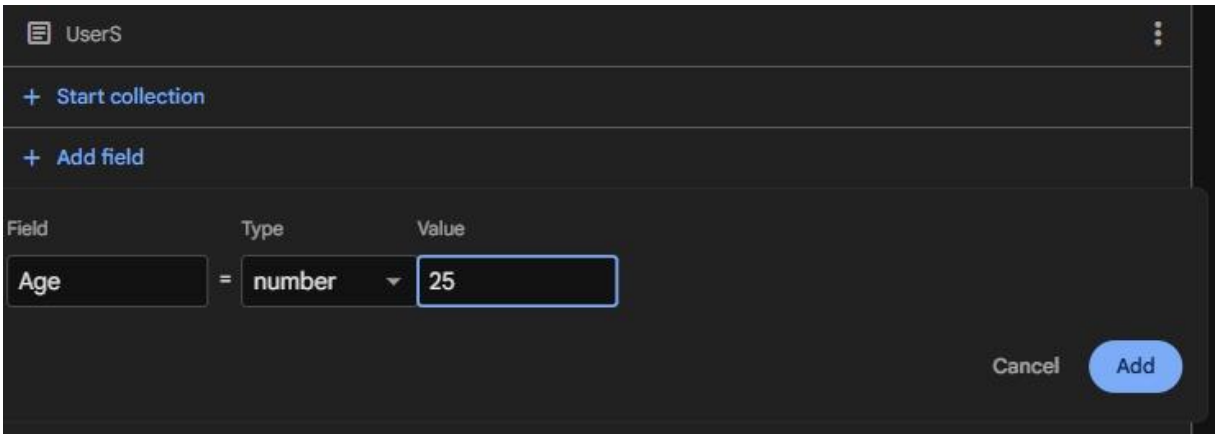
-Click Save to create your first document inside the collection.

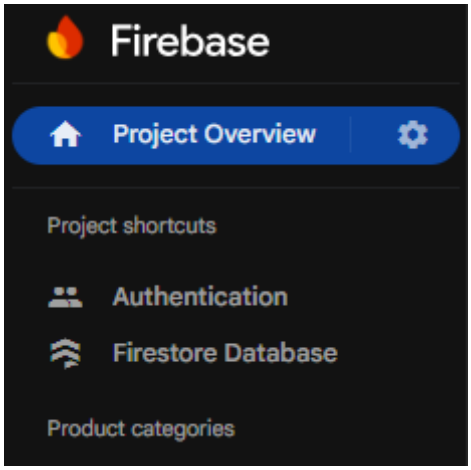


-After creating your collection and first document, you can continue adding new fields at any time.

-This screen allows you to define additional metadata such as:



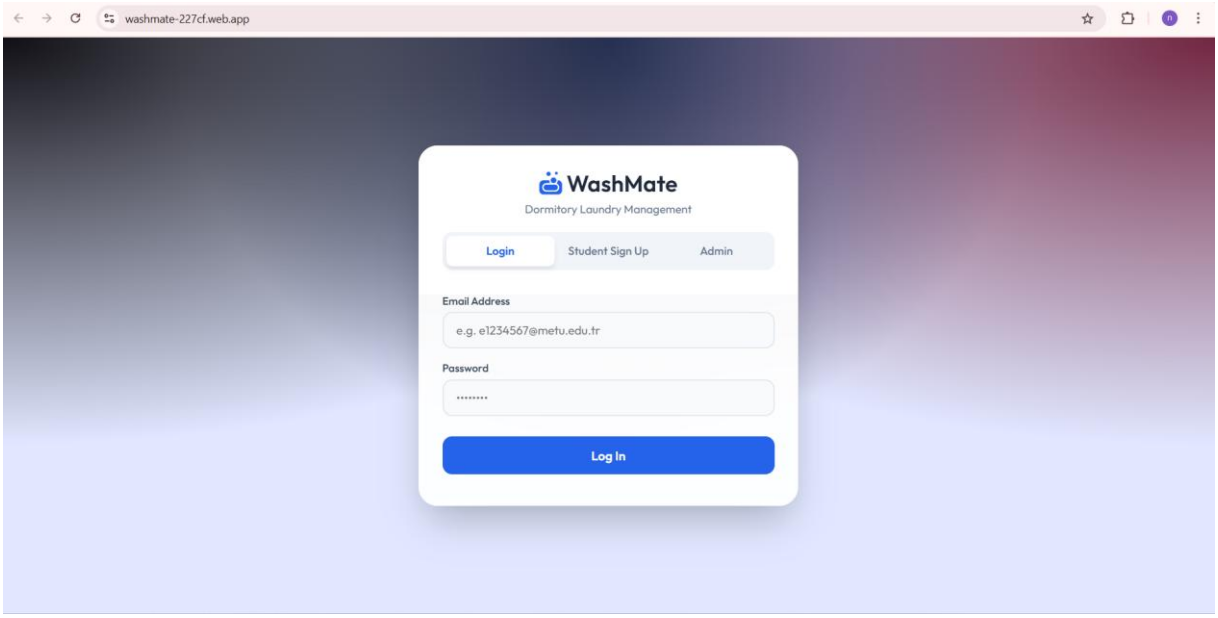




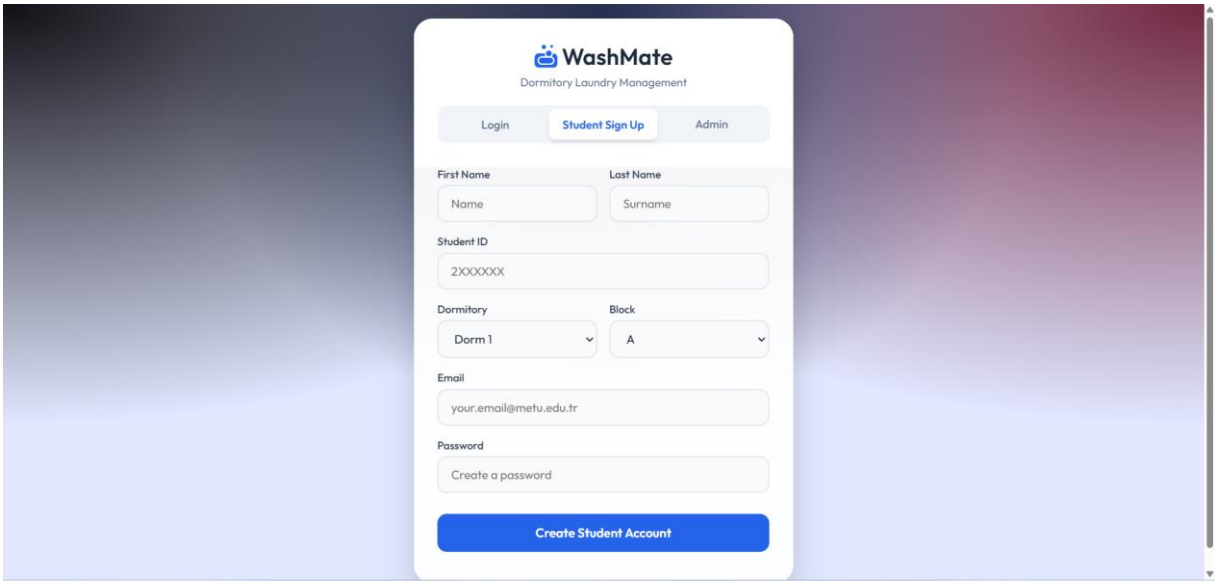
-Once Authentication and Firestore are fully set up, you can always return here to manage your backend.

-These shortcuts let you quickly access user authentication settings and your Firestore database, making it easy to continue building and maintaining your app's server-side features.

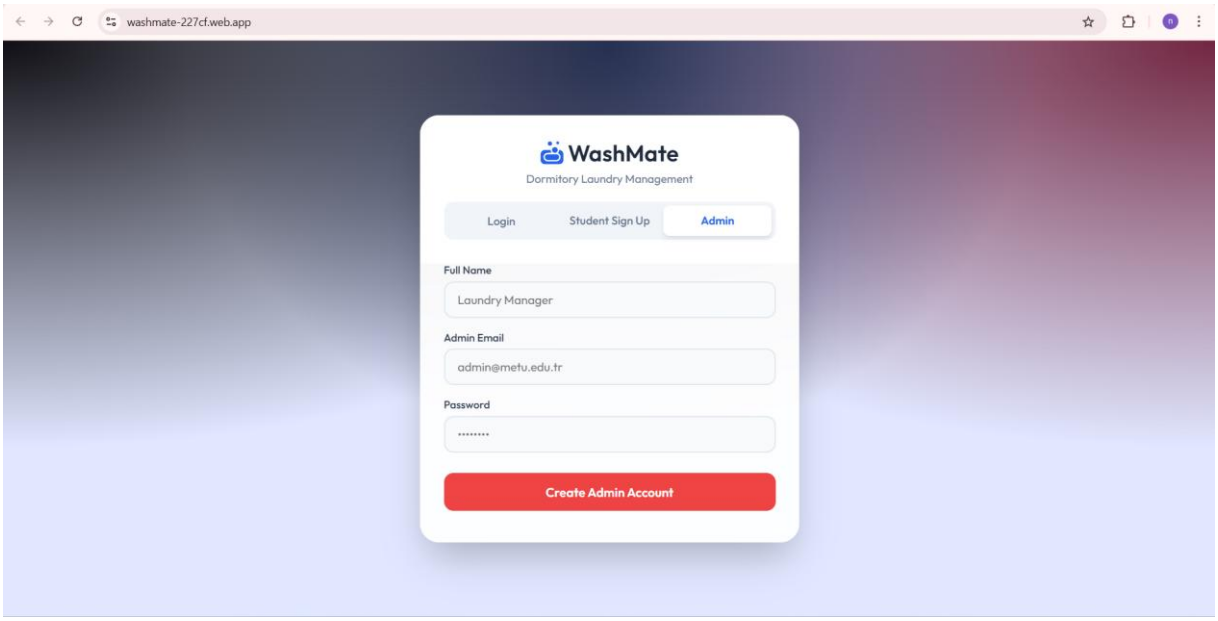
WashMate App Current Version (NAZLICAN):



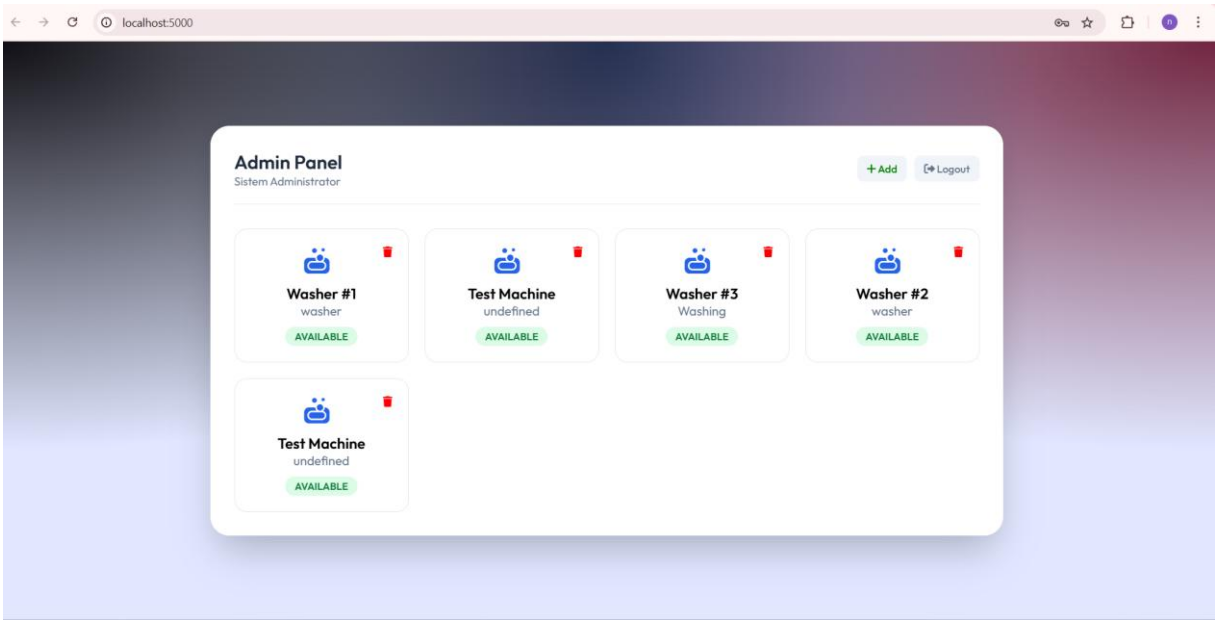
This image is the first screen in our website. Already registered user can login to the website. Also it has buttons for registering next to Login butoon, one for the student sign up and other for Admin signUp.



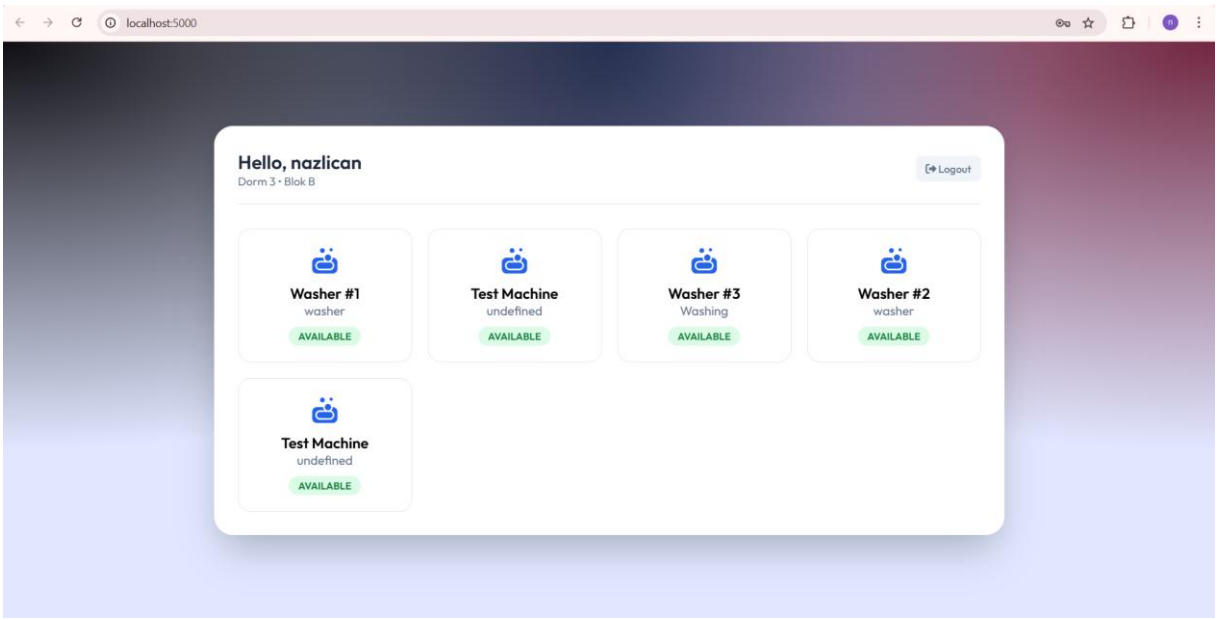
This is the student sign up screen when user selects that, it shows the form to register into the system.



And this is the admin register form. It is again in the first page the user can select that part. As mentioned in before part when user registers into our system, we are succesfully keeping their infromation and role in our Fireastore Database, and after registering they can login.



This part shows the admin panel, here admin can add a machine, delete a existing machine and see the status of the machines. We also checked with database this functionalities are working without a problem.



This image is the user dashboard, user can see the status of the machines.

Milestones Remaining:

Weeks	Milestones	Group Member
Week 11 (01 DEC-08 DEC)	Implementation of the Booking System. In database booking collections should be created and full logic should be implemented.	Nisa
Week 12 (08 DEC-15 DEC)	Research about Cloud functions and notification services and implement them in the system.	Fatih
Week 12 (08 DEC-15 DEC)	Admin analytics and advanced management operations will be implemented. Information page will be added	Nazlıcan
Week 13 (15 DEC-22 DEC)	Create test cases, and test the website according to them. Finalize the project logic.	All Members
Week 14 (22 DEC-29 DEC)	Finalize the stage 3 Report and preparation for the demo.	All Members

Github:

You can reach our repository from the given link below. The repository is public and available:

https://github.com/NazlicanTavis/WashMate.git

REFERENCES

Amazon Simple Storage Service (S3) — Cloud Storage — AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/en/s3/faqs/?nc=sn&loc=7

Amazon Simple Notification Service (SNS) Documentation – AWS. Amazon Web Services, Inc. https://docs.aws.amazon.com/sns/latest/dg/sns-email-notifications.html

Welcome to Flask — Flask Documentation (3.0.X). (n.d.). https://flask.palletsprojects.com/en/3.0.x/

Draw.io. (n.d.). https://www.drawio.com/

StarUML. (n.d.). StarUML. https://staruml.io/