Published using Google Docs Learn more

App Dev I Project Statement - Sept 2025

Updated automatically every 5 minutes

Hospital Management System

Hospitals need efficient systems to manage patients, doctors, appointments, and treatments. Currently, many hospitals use manual registers or disconnected software, which makes it difficult to manage records, avoid scheduling conflicts, and track patient history.

You are required to build a **Hospital Management System (HMS)** web application that allows **Admins, Doctors, and Patients** to interact with the system based on their roles.

Frameworks to be used

These are the mandatory frameworks on which the project has to be built.

- **Flask** for application back-end
- **Jinja2** templating, **HTML**, **CSS** and Bootstraps for application front-end
- **SQLite** for database (**No other database is permitted**)

Note:

- All demos should be possible on your local machine.
- The database must be created programmatically

Published using Google Docs Learn more

App Dev I Project Statement - Sept 2025

Updated automatically every 5 minutes

Roles & Functionalities

1. **Admin (Hospital Staff)**
    a. Admin is the pre-existing superuser of the application
    b. Can add, update, and delete doctor profiles (name, specialization, availability).
    c. Can view and manage all appointments.
    d. Can search for patients or doctors by name/specialization.
2. **Doctor**
    a. Can log in to view assigned appointments.
    b. Can mark a patient's visit as ***completed*** and enter diagnosis & treatment notes.
    c. Can view patient history (previous diagnoses & prescriptions).
3. **Patient**
    a. Can register, log in, and update their

Published using Google Docs Learn more

App Dev I Project Statement - Sept 2025 Updated automatically every 5 minutes

c. Can book, reschedule, or cancel an appointment.

d. Can view their own appointment history and treatment details.

Key Terminologies

1. **Admin (Hospital Staff):** A user with the highest level of access who manages doctors, appointments, and overall hospital data.

2. **Doctor:** A medical professional registered in the system who interacts with patients via the app.

3. **Patient:** A user who seeks medical care and interacts with doctors via the system.

4. **Appointment:** A scheduled meeting between a patient and a doctor for consultation or treatment.

   Attributes:

   1. Patient ID

   2. Doctor ID

   3. Date

Published using Google Docs Learn more

6. Extra fields etc.

5. **Treatment:** A record of medical care provided to a patient during an appointment.

   Attributes:

   1. Appointment ID
   2. Diagnosis
   3. Prescription
   4. Notes.
   5. Extra fields etc.

6. **Department/Specialization:** A field of medical science in which a particular doctor is specialized in

   Attributes:

   1. Department ID
   2. Department Name
   3. Description
   4. Doctors_registered
   5. Extra fields etc.

Note: The above tables and fields are not exhaustive, students can add more tables and fields as per their requirements

Published using Google Docs Learn more

App Dev I Project Statement - Sept 2025

Updated automatically every 5 minutes

Application Wireframe

Hospital Management Application[will be added soon]

Note:

The provided wireframe is intended **only to illustrate the application's flow** and demonstrate what should appear when a user navigates between pages.

- **Replication of the exact views is NOT mandatory.**
- Students are encouraged to work on their front-end ideas and designs while maintaining the application's intended functionality and flow.

Core Features

- **Admin functionalities:**

  a. Admin dashboard must display total number of doctors, patients, and appointments.

  b. Admin should pre-exist in the app i.e. it must be created programmatically after the creation of the database. **[No admin registration allowed]**

  c. Admin can add/update/delete doctor and patient profiles.

  d. Admin can view all upcoming and past appointments.

Published using Google Docs Learn more

details such as name, specialization etc., and also patient info if needed.

g. Admin can remove/blacklist doctors and patients from the system.

- **Doctor functionalities:**

a. Doctor’s dashboard must display upcoming appointments for the day/week.

b. Doctor’s dashboard must show list of patients assigned to the doctor.

c. Doctor's dashboard must have the option to mark appointments as *Completed* or *Cancelled*.

d. Doctors can provide their availability for the next 7 days.

e. Doctors can update patient treatment history like provide diagnosis, treatment and prescriptions.

- **Patient functionalities:**

a. Patients can register and login themselves on the app.

Published using Google Docs Learn more

c. Patients' Dashboard must display availability of doctors for the coming 7 days (**1 week**) and patients can read doctors profiles.

d. It must display upcoming appointments and their status.

e. It must show past appointment history with diagnosis and prescriptions.

f. Patients can edit their profile.

g. Patients can book as well as cancel appointments with doctors.

- **Other core functionalities:**

a. Prevent multiple appointments at the same date and time for the same doctor.

b. Update appointment status dynamically (Booked → Completed → Cancelled).

c. Admin and Patient should be able to search for a specialization or by a doctor's name

Published using Google Docs Learn more

e. Store all completed appointment records for each patient.

f. Include diagnosis, prescriptions, and doctor notes for each visit.

g. Allow patients to view their own treatment history.

h. Allow doctors to view the full history of their patients for informed consultation.

Recommended and/or Optional functionalities

- API resources are created to interact with the users, appointments etc. (Please note: you can choose which API resources to make from the given ones, It is NOT mandatory to create API resources for CRUD of all the components)
- APIs can either be created by returning JSON from a controller (with at least 4 http methods) or using a flask extension like flask_restful
- External APIs/libraries for creating charts, e.g. Chart JS
- Implementing frontend validation on all the form fields using HTML5 form validation or JavaScript
- Implement backend validation within your app's controllers.
- Provide styling and aesthetics to your application by creating a beautiful and responsive front end using simple CSS or Bootstrap **(No**

Published using Google Docs Learn more

App Dev I Project Statement - Sept 2025 Updated automatically every 5 minutes

etc.
- Any additional feature you feel is appropriate for the application

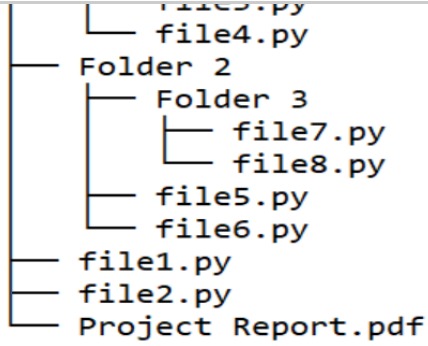
Evaluation

- Students have to create and submit a project report (not more than 5 pages) on the portal, along with the actual project submission
- The report must include the following things;
  - Student details
  - Project details, including the question statement and how you approached the problem statement
  - AI/LLM Declaration
  - Frameworks and libraries used
  - ER diagram of your database, including all the tables and their relations
  - API resource endpoints (if any)
  - Drive link of the presentation video

- **If a student has used any form of AI/LLM for the project, you will need to mention the extent of use in your report, (click here for the doc)**
- **You can find a template to create the project report, (click here for project report demo)**

Possible folder structure:

Published using Google Docs Learn more



- All code is to be submitted on the portal in a single zip file (zipping instructions are given in the project document - Project Doc T32025

- Video Presentation Guidelines (Advised):
    1. A short Intro (not more than 30 sec)
    2. How did you approach the problem statement? (30 sec)
    3. Highlight key features of the application (90 sec)
    4. Any Additional feature(s) implemented other than core requirements (30 sec)

    Note:
    1. The final video **must not exceed 5 – 10 minutes**.
    2. Keeping your video feed on, during recording (like in a screencast) is optional but recommended.

- The video must be uploaded on the student drive with **access to anyone with the link** and the link must be included in the report:
    - This will be viewed during or before the viva, so it should be a clear explanation of your work.
- Viva: after the video explanation, you are required

Published using Google Docs Learn more

live demo.
    - Other questions may be unrelated to the project itself, but are relevant to the course.

Instructions

- This is a live document and will be updated with more details (wireframe)
- We will freeze the problem statement on or before **20/09/2025**, beyond which any modifications to the statement will be communicated via proper announcements.
- The project has to be submitted as a single zip file.