D1

CS 319

2024-25 Fall semester

S3T4 - Code Busters

Umay Dündar 22202573

Elif Ercan 22201601

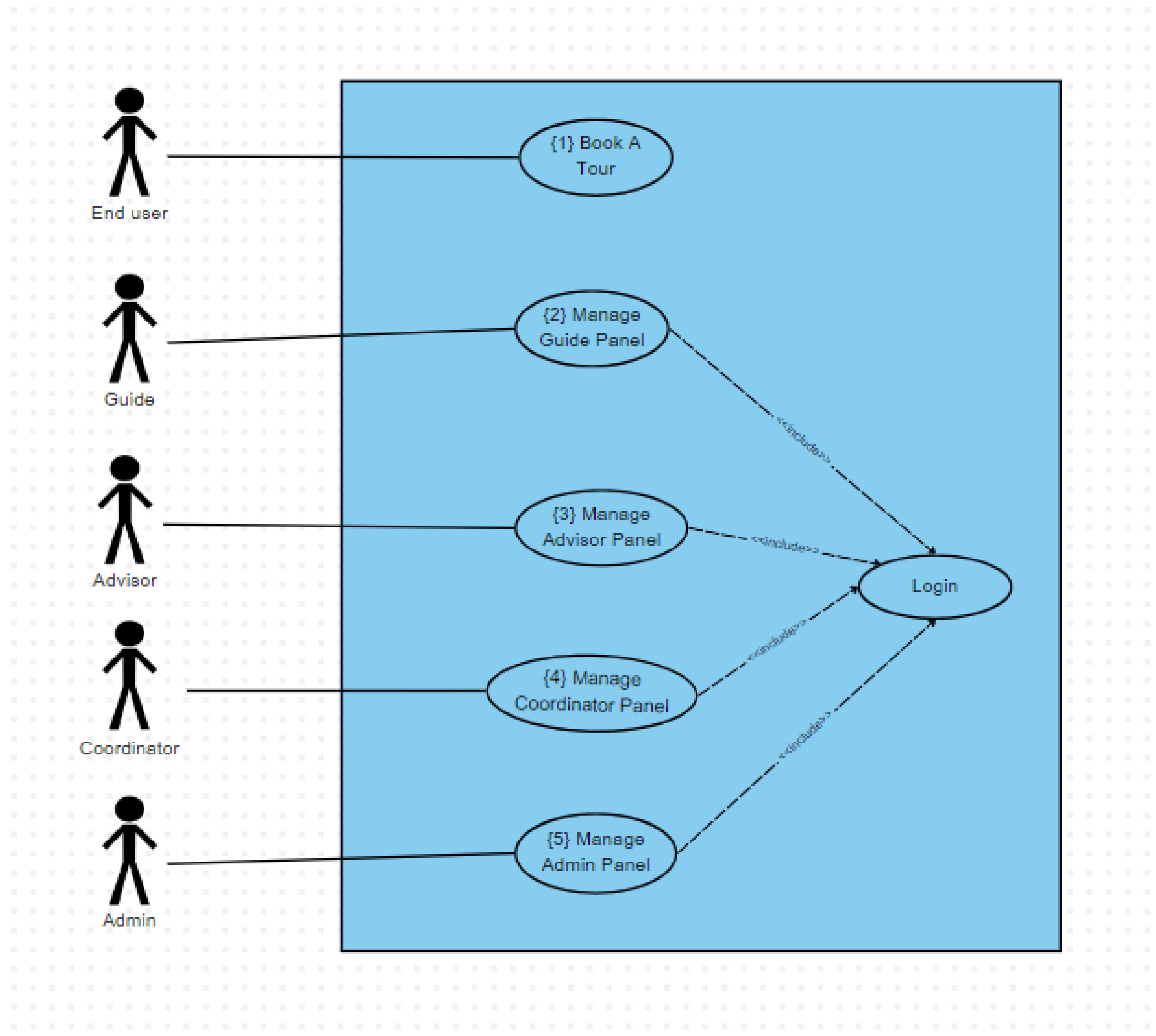
İbrahim Barkın Çınar 22003874

Kemal Onur Özkan 22201820

Contents

1 Use Cases

1.1 Use Case UML Diagram (Level 0)



The details of these {1}, {2}, {3}, {4}, {5} is explained in L1 stage which will be uploaded to moodle individually

2. Non-Functional Requirements

2.1 Security

- Users other than the schools must be authenticated with a username and password combination.
- Passwords will be saved in hashed form in the database.
- Only the admin and coordinator users can create and delete other user accounts, ensuring that outside users are not in the management system.
- Every user in the system has a specialized role, and only the related part of the website is shown to them, which also maintains security.

2.2 Performance

- The website should respond in up to 1 second after a click.
- The web app should load within 3 seconds on standard broadband.
- Search results within the app should appear within 2 seconds.
- Daily, weekly and monthly tour schedules should be retrieved and loaded in less than 1 second.

2.3 Usability

- The website should have a UI with colour coding that is concise and intuitive to use, which gives the page a high learning curve.
- The web app should provide helpful error and info messages to the users.
- Users can access their needs in less than 5 clicks on the webpage.

2.4 Scalability

- The website should be able to host data of over 100 users without losing performance.
- The software should support real-time data passing and retrieving between users who have different profile types, even if there are over 100 users.

2.5 Reliability

- The system should be available at least within the working hours.
- The system should be backed up regularly (like on a weekly basis or monthly basis) to be able to save critical information in a reasonable time (so that the system can be available during working hours).

2.6 Compatibility

- The system should be compatible with the common web browsers.
- The system should work on different computers.

2.7 Accessibility

- The website should be accessible and easy to use, which ensures users with different technical capabilities can easily interact with it.
- All interactive elements should be of a reasonable size, and form inputs should have informative labels that increase the accessibility of the website for outside users.

2.8. Maintainability

- The web app should have clean code with sufficient comments to explain the code, making the future maintainers' jobs easier.
- The code should be modular and well-organized, which will make future code changes, bug fixes and feature additions faster.

2.9 Data Integrity

- Ensure that database data is accurate, consistent, and updated in real-time.
- Data should be backed up in case of emergency.
- Partial manipulation of the database instances is not allowed in most cases to ensure that no data change happens when a data loss occurs.

3. Tech Stack

We will use the **MERN (Mongo, Express, React, Node)** tech stack to create this project. We chose this tech stack because it is commonly used and has many usable libraries and interfaces that can be used in our project.

Frontend: React.js

API: Express

Backend: Node.js

Database: MongoDB